Sparse Modeling in Classification, Compression and Detection

A Thesis
Presented to
The Academic Faculty

by

Jihong Chen

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Industrial and Systems Engineering
Georgia Institute of Technology
July 2004

Sparse Modeling in Classification, Compression and Detection

A Thesis
Presented to
The Academic Faculty

by

Jihong Chen

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Industrial and Systems Engineering
Georgia Institute of Technology
July 2004

Sparse Modeling in Classification, Compression and Detection

Approved by:

Prof. Xiaoming Huo, Advisor

Prof. Jye-Chyi Lu

Prof. Kwok-Leung Tsui

Prof. Brani Vidakovic

Prof. Xiaoping Hu

Date Approved: 22 June 2004

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Dr. Xiaoming Huo for his great help and support during my graduate studies. His patience and insight make it a pleasure to work with him. Thanks are also due to other members of my committee, Professor Jye-Chyi Lu, Xiaoping Hu, Kwok-Leung Tsui, Brani Vidakovic and Guotong Zhou. On a more personal level, I would like to thank my close friends, Qian Li, Xiantian Dai, Ping Jing and Yun Xing. I am also grateful to my family for their love and support throughout. Finally, my husband, Wuqin Lin, has been a tremendous source of encouragement in the past three years. His genius and creativity has given me great help in my research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The principal focus of this thesis is the exploration of sparse structures in a variety of statistical modelling problems. While more comprehensive models can be useful to solve a larger number of problems, its calculation may be ill-posed in most practical instances because of the sparsity of informative features in the data. If this sparse structure can be exploited, the models can often be solved very efficiently.

The thesis is composed of four projects. Firstly, feature sparsity is incorporated to improve the performance of support vector machines when there are a lot of noise features present. The second project is about an empirical study on how to construct an optimal cascade structure. The third project involves the design of a progressive, rate-distortion-optimized shape coder by combining zero-tree algorithm with beamlet structure. Finally, the longest run statistics is applied for the detection of a filamentary structure in two-dimensional rectangular region.

The fundamental ideas of the above projects are common — to extract an efficient summary from a large amount of data. The main contributions of this work are to develop and implement novel techniques for the efficient solutions of several difficult problems that arise in statistical signal/image processing.

CHAPTER I

INTRODUCTION

1.1 Motivation

The goal of modeling is to construct a model that best accounts for observed data. Statistical models of natural images underlie a variety of applications in image processing and computer vision, such as image coding, target detection, pattern recognition, etc. The goodness of a model is often characterized by its performance in prediction, interpretability, computational efficiency and ease for knowledge extraction. These properties depend crucially on the ability to construct appropriate sparse models by the modeling process. In the following, we give some applications showing the importance of sparse modeling.

In machine learning, the objective of modeling from data is not that the model simply fits the training data well. Rather, a model is evaluated by its generalization capability and interpretability. There exists a vast amount of works in the area of sparse modeling (e.g, [64],[13]). Recently, a well-known sparse kernel modeling algorithm is the support vector machine (SVM) method [64], which is widely regarded as the state-of-the-art technique for regression and classification applications. The formulation of SVM embodies the structural risk minimization principle, thus combining excellent generalization properties with a sparse model representation. Despite these attractive features and many good empirical results obtained using the SVM method, researchers have realized that the ability for the SVM method to produce sparse models has perhaps been overstated. The problem of performing feature selection for SVMs has been investigated in several recent papers (e.g., [17, 70]). Our motivation is twofold. First, we aim to derive a learning machine based directly on optimizing model generalization capability. We also want the feature selection process to be automatic without the need for the user to specify the number of features. The usual feature selection approaches in the literature cannot achieve these objectives.

Hierarchical-structure-based methodologies have been very popular in recent years. For example, CART and MARS [13], C4.5 [55], perception trees [10], quad-tree based image coding [57, 60], pyramid [16], etc. Specially, consider a hierarchy of classifiers such that at each level there are exactly two branches, one of which is terminal: it has no subsequential classifier. This sparse tree structure is known as a *cascade.* Cascades are the fundamental structures of many successful techniques in various applications (such as [23], [39, 38], [66], [37]). Therefore, a systematic study on how to search for the optimal cascade will be interesting, and may be integrated into existing methods to improve their performances.

The compression of images is important in both transmission and storage. The key

idea is to make use of the particular sparse structure in an image matrix. In the area of image compression, contour coding has been an important topic because of its special applications in video coding and artificial images. Chain coding is the earliest contour coding methodology that was attributed to [28, 29]. The key idea of a chain coder is to code, at each step, multiple and aligned pixels (that form a line segment) instead of one nearest neighbor. Alternatives other than straight line segment have been explored as basic elements. These works include discrete arcs [11], polygonal curves [46], and a lot more. Researchers have noticed that it is not necessary to exactly follow a curve given by a discrete image. Methods taking into account of the trade-off between the fidelity and efficiency of a contour codec are studied in [59]. More interestingly, the idea of using multiscale structure (i.e. pyramid) to code a contour has been studied by several researchers, e.g. [48, 50]. Despite all the existing work, the idea of zero-tree coding, which was successfully established in wavelet coding and adopted in JPEG 2000, has *not* been adopted to code curves.

Detecting the presence of a filament (or a filament-like feature) in an image is a fundamental problem in many applications, such as medical image processing, target recognition, satellite image analysis, and many more. Recently, there has been significant advances in mathematical statistics on this problem. Namely, in [6], it is proven (as one of many cases) that when the underlying features is a digitized line segment, when the amplitude of the signal residing on the underlying feature is strong enough, there is a statistically efficient means to reliably detect it. On the other hand, if the amplitude of the signal is not strong enough, there is no method that can detect it. Moreover, there is an algorithm that achieves the lowest possible order of complexity. The fundamental tools used are introduced in [22]. To detect a filamentary feature, which is more likely to be *curvilinear*, the previous result, which requires *straight* line segment, is not strong enough. In a working paper [7], for point processes, it is shown that the number of points on a smooth function is governed by certain laws. This result gives the possibility to calculate the detectability of an underlying curvilinear feature in a point process. Even though the result from point processes can *not* be directly applied to imagery data, the developed techniques – covering and the related probabilistic analysis – can be adapted for the digital problem.

1.2 Contributions

The main contributions of this work are to sensibly combine existing ideas along with new ones for the efficient solutions of several difficult problems that arise in statistical signal/image processing. The thesis is composed of four main chapters.

Chapter 2 is about how to incorporate feature sparsity to improve the performance of support vector machines. We first show that the VC dimension of separating hyperplanes decreases with the dimensionality of the feature subspace. Motivated by this observation,

we propose a new approach *Penalized Support Vector Machine*, which includes a penalization function to suppress the dimensionality. An analysis of the dual problem leads to the penalized least square problem which has been extensively discussed by [27]. Our proposed method does feature selection while training SVMs simultaneously. Simulation results confirm the feasibility of our approach in performing feature selection and demonstrate an improvement of generalization performance over SVMs using all features.

In Chapter 3, a hierarchical classifier (cascade) is proposed for target detections. In building an optimal cascade, three heuristics are considered: (1) frontier following approximation, (2) controlling error rates, and (3) weighting. Simulations on synthetic data with various underlying distributions are carried out. It is found that weighting heuristic is optimal in both computational complexity and error rates. We initiate a systematic comparison of several potential heuristics that can be utilized in building a hierarchical model. The weighting algorithm is applied to an IR data set. It is found that it outperforms some state-of-the-art approaches that utilize the same type of simple classifiers.

In Chapter 4, a multiscale coder for curves and boundaries is presented. It utilizes a multiscale structure – beamlets – that is designed for lines and curves. The coder is composed of three main components: (1) a rate-distortion optimized beamlet-based representation, (2) a zero-tree coding from a beamlet representation to a symbol stream, and (3) an entropy coder. This coder is named "JBEAM". Due to its multiscale property, we utilized zero-tree coding to make it progressive. The derived coder has low order of numerical complexity. Simulations demonstrate an advantage over the state-of- the-art industrial standard: JBIG 2. Variations and potential improvements of this method are discussed. We hope this work will inspire more activities in this line of research for curve coding.

Chapter 5 focuses on the design of a general framework for detection in imagery data – Significance Run Algorithm (SRA). The detection method can detect the presence of an underlying curvilinear feature with the lowest possible strength that are still detectible in theory. The 'most powerful' detection can be realized, for a set of specifically organized underlying objects. It is shown that by assigning an hierarchy to the alternatives, one can nearly realize the most powerful detection under certain conditions. Simulation results on synthetic data will be reported to illustrate its effectiveness in finite digital situations.

1.3 Outline of Thesis

This thesis is organized as follows:

- Chapter 2 provides a feature selection approach for Support Vector Machines. We first prove that the VC dimension of separating hyperplanes decreases with the dimensionality of the feature subspace. This motivates a new approach *Penalized Support Vector Machine*, which includes a penalization function to suppress the dimensionality. An

analysis of the dual problem leads to the traditional penalized least square problem. Some statistical properties are discussed. Our proposed method does feature selection while training SVMs simultaneously.

- Chapter 3 gives an empirical study of several potential heuristic methods in building an optimal cascade. Three heuristics are considered: (1) frontier following approximation, (2) controlling error rates, and (3) weighting. Simulations on synthetic data with various underlying distributions are carried out for a systematic comparison. A range of discussions regarding the implications and the promises of the cascade architecture, as well as techniques that can be integrated into this framework are provided.

- Chapter 4 proposes a progressive compression scheme for arbitrary shapes. Beamlet transform is combined with a zerotree algorithm to create progressive coder. The hierarchical structure of beamlets facilitates an easy algorithm for a *rate-distortion-optimized* coder.

- Chapter 5 is concerned with detecting the presence of a filament in images. We focus on the design of a general framework for detection procedures – Significance Run Algorithm (SRA). It is shown that by assigning an hierarchy to the alternatives, one can realize the nearly most powerful detection under certain conditions. Specifically, methods to solve detection problems in noisy images and textured images are proposed.

- Chapter 6 provides a review of the overall contributions of this thesis, and provides some ideas for future work.

CHAPTER II

FEATURE SPARSITY

This chapter studies how to incorporate feature sparsity to improve the generalization performance of support vector machines. In section 2.2, we show that the VC dimension of separating hyperplanes has an upper bound depending on the dimensionality of feature subspace. In section 2.3, the new approach PSVM is introduced, and its properties are discussed. A simple algorithm is proposed for implementation. Simulation studies are conducted to evaluate the feasibility of the new approach in section 2.4. Conclusions are given in Section 2.5.

2.1 Introduction

The problem of feature selection is to use a small number of features while preserving or improving the generalization performance. Traditional feature selection methods conduct a heuristic search for a good subset according to some evaluation function. Based on the dependence of evaluation criteria on the induction algorithm, there are two categories: the filter method and the wrapper method [47]. The filter method is used as a preprocessing step to the induction algorithm. In the wrapper method, on the other hand, feature subset search is conducted using the induction algorithm itself as part of the evaluation functions. The subset selection methods, although of practical use, suffer from several drawbacks including instability [12]. Statisticians have resorted to more continuous methods in an attempt to do feature selection automatically. [27] proposed penalized likelihood approaches to handle variable selection problem in linear regression analysis. Their approach is based on penalized least square. The penalty functions are selected so that the resulting estimator takes the form of a thresholding rule. In this way, subset selection algorithm is automatically embedded and exists as a wrapper around the induction algorithm. The approach retains good features of subset selection while avoiding the instability brought by discrete search process.

In recent years, Support Vector Machines (SVMs) have been successfully applied to a large number of classification problems [64, 14, 52]. However, in the book by [35, Chap. 12], it was stated that the kernel cannot adapt itself to concentrate on subspaces and support vector classifier does poorly when noise features are present. The problem of performing feature selection for SVMs has been investigated in several recent papers. Among them, a representative work by [17] and [70] applied a wrapper method which is based on minimizing the bound on leave-one-out error by a gradient descent algorithm. The method minimizes

a function that has the form $R^2(\sigma)W^2(\alpha,\sigma)$, where σ is a vector made by 0's and 1's and α is a real vector. Since σ is discrete, minimizing function $R^2(\sigma)W^2(\alpha,\sigma)$ is conceptually a combinatorial optimization problem, which can be computationally very expensive. [70] proposed several numerical approximation strategies to overcome this problem.

In this chapter, we propose a new approach via penalized least square to incorporate feature selection into SVMs. Our analysis is focus on classification problems. We first explore how feature selection helps improve the generalization performance from the view of the Vapnik-Chervonenkis(VC) dimension [64]. We show that the VC dimension of hyperplanes defined in feature subspace has an upper bound proportional to the dimensionality of the subspace. This observation motivates a new learning machine *Penalized Support Vector Machine* (PSVM), which differs from conventional SVM by including an additional penalty function to suppress the dimensionality of the feature space. We find, interestingly, the dual form of PSVMs leads to the penalized least square problem which has been discussed in details by [27]. The statistical properties of the new learning machine are studied. Feature selection is simultaneously performed while training SVM.

Compared to existing methods, the new approach has several advantages. First, the PSVM attempts to present a unified framework: PSVM leads to a well defined optimization problem, even though an algorithm parameter is embedded to control the trade-off between model goodness of fit and the penalty. Secondly, PSVM can be solved by using standard numerical software. Thirdly, feature selection is automatically embedded in the algorithm. While in the gradient descent method, the number of useful features has to be pre-given as a known information.

2.2 Feature Selection and the VC Dimension

In binary classification problems, we are given training data $\mathbf{x}_i = (x_i^1, x_i^2, \ldots, x_i^n) \in \mathbb{R}^n, i = 1, \ldots, l$, with corresponding class labels $y_i \in \{-1, 1\}, i = 1, \ldots, l$. For hyperplane classifiers, the problem of selecting a feature subset of size m ($m < n$) is equivalent to find hyperplanes defined as follows

$$\mathcal{S} = \{f(\mathbf{x}) = \beta \cdot \mathbf{x} + \beta_0 : \quad \mathbf{x}, \beta \in \mathbb{R}^n,\ \|\beta\|_0 \leq m\}, \tag{1}$$

where $\|\cdot\|_0$ counting the nonzero elements in the vector. Here the functions in set $\mathcal{S}$ is referred to as m-hyperplanes because each function uniquely corresponds to a hyperplane defined in some m dimensional subspace.

In machine learning theory, one of the most widely used capacity concept is the Vapnik-Chervonenkis(VC) dimension [64]. It has been established that the generalization error of classifiers is bounded by the sum of the empirical error and an increasing function of the VC dimension. The VC dimension is a property of a set of functions $\{f(\alpha)\}$, where α denotes a generic set of parameters and a choice of α specifies a particular function. For binary

classification case, $f(\mathbf{x}, \alpha) \in \{-1, 1\} \forall \mathbf{x}, \alpha$. Now if a given set of ℓ points can be labelled in all possible 2^ℓ ways, and for each labelling, a member of the set $\{f(\alpha)\}$ can be found which correctly assigns those labels, we say that that set of points is shattered by that set of functions. The VC dimension for the set of function $\{f(\alpha)\}$ is defined as the maximum number of training points that can be shattered by $\{f(\alpha)\}$. Note that, if the VC dimension is h, then there exists at least one set of h points that can be shattered, but in general it will not be true that every set of h points can be shattered.

For hyperplane classifiers in $\mathbf{R}^n$, the VC dimension is equal to $n + 1$. In the following, we provide a theorem for an upper bound on the VC dimension of $\mathcal{S}$.

Theorem 1. *The VC dimension h of $\mathcal{S}$ is bounded by the inequality*

$$h \leq \min\{(2m + 1) \log_2(n + 1), n + 1\}. \tag{2}$$

Proof:

For the ease of exposition, we define

$$\mathcal{K} = \{\mathbf{k} = (k_1, \ldots, k_m) : 1 \leq k_1 < k_2 < \cdots < k_m \leq n\},$$

and index each m-dimensional feature subspace by a $\mathbf{k} \in \mathcal{K}$. For each $\mathbf{k} \in \mathcal{K}$,

$$\mathcal{S}_\mathbf{k} = \{f(\mathbf{x}) = \beta \cdot \mathbf{x} + \beta_0 : \quad \mathbf{x}, \beta \in \mathbb{R}^n, \beta_k = 0, k \notin \mathbf{k}\}$$

is defined as the set of m-hyperplanes with each corresponding to a hyperplane defined in the feature subspace $\mathbf{k}$. Obviously $\mathcal{S} = \bigcup_{\mathbf{k} \in \mathcal{K}} \mathcal{S}_\mathbf{k}$, and it is easy to verify that the VC dimension of $\mathcal{S}_\mathbf{k}$ equals the VC dimension of the set of oriented hyperplanes defined in feature subspace $\mathbf{k}$ which is $m + 1$. Given a set $\mathcal{F}$ of functions, the *shatter function* $\pi(\mathcal{F}, \ell)$ is the maximum number of distinct separations of any ℓ points in $\mathbb{R}^n$ that can be separated by the functions in $\mathcal{F}$. By the shatter function lemma [65], it follows, from the fact that the VC dimension of $\mathcal{S}_\mathbf{k}$ is $m + 1$, that

$$\pi(\mathcal{S}_\mathbf{k}, \ell) \leq \sum_{i=0}^{m+1} \binom{\ell}{i} \leq \ell^{m+1}. \tag{3}$$

Now let h be the VC dimension of $\mathcal{S}$, then by the definition,

$$2^h = \pi(\mathcal{S}, h) \leq \sum_{\mathbf{k} \in \mathcal{K}} \pi(\mathcal{S}_\mathbf{k}, h). \tag{4}$$

The last inequality holds because $\mathcal{S} = \bigcup_{\mathbf{k} \in \mathcal{K}} \mathcal{S}_\mathbf{k}$. Then it follows from (3) and (4) that

$$2^h \leq \binom{n}{m} h^{m+1} \leq n^m h^{m+1} < (n + 1)^{2m+1}. \tag{5}$$

Note that there are in total $\binom{n}{m}$ m-dimentional feature subspaces and that $h \leq n+1$ because $\mathcal{S}$ is a subset of the set of oriented hyperplanes in $\mathbb{R}^n$, which has VC dimension of $n + 1$. The results then follow from (5).

This upper bound, although being very loose actually, illustrates the idea that the VC dimension of hyperplanes decreases with the dimensionality of feature subspace. This gives an explanation why with similar empirical performance, the classifier which uses fewer features usually tends to have better generalization performance. This observation provides the motivation of our approach. We believe there exists a tighter bound, but it is not the purpose of this chapter to seek for it.

2.3 Penalized Support Vector Machines

In this section, based on the observation in last section, we introduce a new approach to perform feature selection for Support Vector Machines. The new approach includes an additional penalty function into SVMs to suppress the dimensionality. A study of the dual formulation leads to the penalized least square problem. Our analysis remains in linear space. It can be naturally extended to nonlinear feature space by simply replacing $\mathbf{x}$ with the corresponding mapping function $\Phi(\mathbf{x})$.

2.3.1 Support Vector Machines

[64] showed that the VC dimension restricted to a particular training data set can be bounded in terms of their margin, measured by $\|\beta\|^2$. This is the key motivation of Support Vector Machines [64], in which the empirical error and the margin are linearly combined in one overall function to be minimized.

For simplicity, we introduce an additional dull dimension: dimension 0 and set $x^0 = 1$ for all samples. The general hyperplane classifiers are expressed as

$$f(\mathbf{x}) = \beta \cdot \mathbf{x} = \sum_{j=0}^{n} \beta_j x^j. \tag{6}$$

SVMs can be written as the following regularized function estimation problem

$$\min_{\beta} \frac{1}{l} \sum_{i=1}^{l} [1 - y_i f(\mathbf{x}_i)]_+ + \lambda \|\beta\|^2, \tag{7}$$

where the subscript "+" indicates positive part and the parameter λ controls the trade-off.

2.3.2 Penalized Support Vector Machines

In this section, we describe an approach to perform feature selection within support vector machines. We have known that incorporating feature selection implies that there are two main intertwining factors in determining the VC dimension: the dimensionality of feature subspace as well as the margin. It is very difficult to model the true underlying relationship between these two factors. From computational consideration, we simply use linear

combinations as an approximation, and consider the following objective function,

$$\min_{\beta} \frac{1}{l}\sum_{i=1}^{l}[1-y_i f(\mathbf{x}_i)]_+ + \lambda_1\|\beta\|^2 + \lambda_2\|\beta\|_0. \tag{8}$$

In the formulation, λ_1 and λ_2 are two trade-off parameters. A good approximation can be reached by adjusting the two parameters.

We look for continuous penalty functions to approximate the discontinuous step function $\|\beta\|_0$ in the formulation (8). These penalty functions must be able to suppress the dimensionality. Some other conditions will be discussed later. We call the unified approach *Penalized Support Vector Machine* (PSVM).

To be consistent with conventional SVM, we write the PSVMs in the following formulation

$$\begin{array}{ll} \min\limits_{(\beta,\xi)} & \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{l}\xi_i + \lambda\sum_{j=0}^{n}p_j(|\beta_j|) \\ s.t. & y_i(\beta\cdot\mathbf{x}_i) \geq 1-\xi_i,\ \xi_i \geq 0,\ i=1,\cdots,l. \end{array} \tag{9}$$

Again, C and λ are two trade-off parameters. When $\lambda = 0$, PSVMs degenerates to SVMs.

2.3.3 PSVM Dual

Without prior knowledge, we assume that the penalty functions for all coefficients are the same: $p_j(|\cdot|) = p(|\cdot|)$. Also we denote $\lambda p(|\cdot|)$ by $p_\lambda(|\cdot|)$. The Lagrange function of (9) is as follows

$$\begin{aligned} L_P &= \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{l}\xi_i + \sum_{j=0}^{n}p_\lambda(|\beta_j|) - \sum_{i=1}^{l}\alpha_i(y_i\beta^T\mathbf{x}_i - 1 + \xi_i) \\ &= \sum_{i=1}^{l}\alpha_i + \frac{1}{2}\sum_{j=0}^{n}\beta_j^2 + \sum_{j=0}^{n}p_\lambda(|\beta_j|) - \sum_{j=0}^{n}(\sum_{i}^{l}\alpha_i y_i x_i^j)\beta_j + \sum_{i=1}^{l}(C-\alpha_i)\xi_i, \end{aligned} \tag{10}$$

where $\alpha_i, \xi_i \geq 0$, and we maximize w.r.t. α and minimize w.r.t. $\beta,\ \xi$. Define

$$\varphi(\beta,\alpha) = \frac{1}{2}\sum_{j=0}^{n}\beta_j^2 + \sum_{j=0}^{n}p_\lambda(|\beta_j|) - \sum_{j=0}^{n}(\sum_{i=1}^{l}\alpha_i y_i \mathbf{x}_i^j)\beta_j, \tag{11}$$

The dual problem can be written as

$$\begin{array}{ll} \max\limits_{\alpha} & \sum_{i=1}^{l}\alpha_i + \varphi^*(\alpha) \\ s.t. & 0 \leq \alpha_i \leq C, \end{array} \tag{12}$$

where $\varphi^*(\alpha) = \min_\beta \varphi(\beta,\alpha)$. Denote the solutions to the above dual problem as α^s and β^s. The resulting hyperplane is

$$f^s(\mathbf{x}) = \sum_{j=0}^{n}\beta_j^s x^j. \tag{13}$$

2.3.4 Selection of Penalty Function

To solve for (13), we have to solve the subproblem of minimizing $\varphi(\beta, \alpha)$ for given α, which can be decomposed componentwise. We can write each decomposed subproblem as

$$\min_{\beta_j} \ \frac{1}{2}\beta_j^2 - z_j\beta_j + p_\lambda(|\beta_j|), \tag{14}$$

where $z_j \equiv \sum_{i=1}^{l} \alpha_i y_i x_i^j$. Note that the solution of β_j to the above problem depends on α in the form of z_j. Therefore we denote the optimal solution to the problem (14) as

$$\phi(z_j) = \operatorname*{argmin}_{\beta_j} \frac{1}{2}\beta_j^2 - z_j\beta_j + p_\lambda(|\beta_j|). \tag{15}$$

We notice that this is exactly the penalized least square problem which has been extensively discussed by [27]. For SVMs, $p_\lambda(|\theta|) = 0$ and $\phi(z_j) = z_j$. To embed feature selection in SVMs, we have to pick penalty functions so that the resulting estimator takes the form of a thresholding rule. In the following, we list three penalty functions and the corresponding optimal solutions to (14).

1. The L_1 penalty: $p_\lambda(|\theta|) = \lambda|\theta|$, and the solution is a soft thresholding rule $\phi(z_j) = sign(z_j)(z_j - \lambda)_+$.

2. The hard thresholding penalty: $p_\lambda(|\theta|) = \lambda^2 - (|\theta| - \lambda)^2 I(|\theta| < \lambda)$, and the solution is a hard thresholding rule $\phi(z_j) = z_j I(|z|_j > \lambda)$.

3. Smoothly Clipped Absolute Deviation Penalty (SCAD):

$$p'_\lambda(\theta) = \lambda \left\{ I(\theta \le \lambda) + \frac{a\lambda - \theta)_+}{(a-1)\lambda} I(\theta > \lambda) \right\} \text{ for some } a > 2 \text{ and } \theta > 0,$$

   and the solution is

$$\phi(z_j) = \begin{cases} sgn(z_j)(|z_j| - \lambda), & |z_j| \le 2\lambda; \\ \{(a-1)z_j - sgn(z_j)a\lambda\}/(a-2), & 2\lambda < |z_j| \le a\lambda; \\ z_j, & |z_j| > a\lambda. \end{cases}$$

We claim the new machine should have the following four properties.

1. *Convexity:* The minimization problem (14) is convex.

2. *SVM consistency:* We suppose that SVMs provide a good estimate for β_j's when the underlying true coefficients are large. So for large $|\beta_j|$, we expect the solution to be consistent with standard SVM. The first derivative of (14) with respect to β_j is $sign(\beta_j)\{|\beta_j| + p'_\lambda(|\beta_j|)\} - z_j$. Obviously, $p'_\lambda(|\beta_j|) = 0$ for large $|\beta_j|$ is a sufficient condition for SVM consistency.

3. *Feature sparsity:* The resulting estimator is a thresholding rule so that small estimated coefficients are set to zero. $\min_{|\theta|>0} \ |\theta| + p'_\lambda(|\theta|) > 0$ is a sufficient condition for sparsity.

4. *Input continuity:* The resulting machine is continuous in the input data to avoid model instability. The sufficient and necessary condition for input continuity is that the minimum of the function $|\theta| + p'_\lambda(|\theta|)$ is attained at 0.

SCAD penalty satisfies all four requirements for convexity, SVM consistence, feature sparsity, and input continuity. The L_1 penalty does not satisfy SVM consistency and the hard thresholding penalty function does not satisfy input continuity. See [27] for more detailed analysis.

2.3.5 Subgradient Algorithm

We apply subgradient method [4] to solve the dual problem (11), and the algorithm is as follows.

1. Initialize $k = 0$, $\alpha^0 = 0$. Given maximum number of loops $kmax$.

2. Set $z^k = \sum_{i=1}^{l} \alpha_i^k y_i x_i$, and $\beta_j^{k+1} = \phi(z_j^k)$.

3. For each i, set $\Delta_i^{k+1} = 1 - y_i \beta \mathbf{x}_i$, $\rho^{k+1} = 1/(k+1)$ and $\alpha' = \alpha_i^k + \rho^{k+1}\Delta_i^{k+1}$, and update
$$\alpha_i^{k+1} = \begin{cases} 0 & \text{if} \quad \alpha' < 0; \\ \alpha' & \text{if} \quad 0 \le \alpha' \le C; \\ C & \text{if} \quad \alpha' > C. \end{cases}$$

4. $k = k + 1$. Loop until $\max_i |\alpha_i^{k+1} - \alpha_i^k| \le \epsilon$ or $k = kmax$.

The algorithm is easy to implement and the convergence is ensured. More efficient algorithms can be developed. For instance, the objective function in the primal PSVM is quadratic-like, so we may approximate it by quadratic functions and use Newton methods.

2.4 Experiments

In the experiments reported below, the tuning parameters (C, λ) are chosen by 5-fold cross-validation. The datasets are normalized before processing. All experiments are conducted using MATLAB codes.

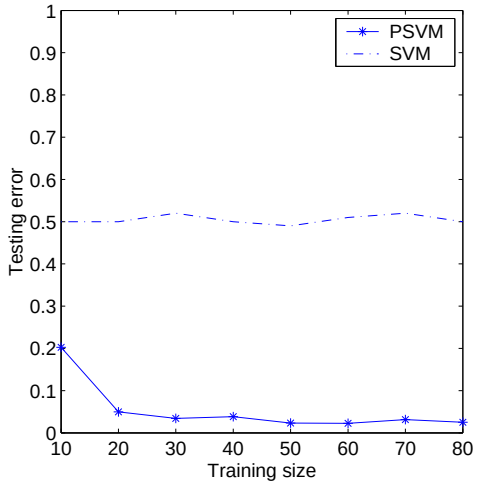


Figure 1: A comparison of PSVM and SVM on the Toy data

Table 1: Results for toy data: Average number of features selected for different training sizes

Training size	10	20	30	40	50	60	70	80
# features	4.4	5.8	6.3	5.2	5.8	5.6	6.3	5.4

2.4.1 Toy data

First, we assess the performance of the algorithm on the linear example used in [17] and [70]. The experiment settings are repeated as follows. The prior probabilities of two classes are the same. The inputs have 202 dimensions. The first three dimensions $\{x^1, x^2, x^3\}$ are drawn from $x^j = yN(j, 1)$ with probability 0.7 and from $x^j = N(0, 1)$ with probability 0.3. The next three dimensions $\{x^4, x^5, x^6\}$ are drawn as $x^j = N(0, 1)$ w.p. 0.7 and from $x^j = yN(j - 3, 1)$ w.p. 0.3. All other dimensions are noises $x_j = N(0, 20), j = 7, \ldots, 202$.

We generate 500 samples for testing. The average testing error rate are calculated over 10 runs of experiments for various training set sizes. A comparison of the results of SVM and PSVM are shown in Figure 1. SVM uses all 202 features and achieves about 50% error rate, while PSVM shows promising performance comparable to the ones obtained by the gradient descent method by [17] and [70]. Moreover, in their method, the number of useful features was provided as a prior information, whereas in our method, it is obtained automatically. In Table 1, we report the average numbers of features selected by PSVM, which is very close to the true number of relevant features.

2.4.2 UCI datasets

We also test our algorithm on six datasets from UCI Machine Learning Repository [51]. Brief descriptions of the datasets are given as follows.

- The Wisconsin Prognostic Breast Cancer database consists of 198 samples with 32 features. The problem is to classify patients with non-recurred cancer against those with recurred cancer. There are 151 non-recur samples and 47 recur samples.
- The Ionosphere database consists 351 instances with 33 continuous features.
- The Image Segmentation dataset consists of 2100 training samples and 210 testing samples with 19 features. These samples were drawn randomly from a database of 7 outdoor images: brickface, sky, foliage, cement, window, path, grass. We consider the problem of classifying the first three images against the other four.
- The Pen-based Recognition of Handwritten Digits dataset consists of 16 continuous features. It is a 10-class problem and we try to classify 6's against 9's. There are 1439 training samples and 672 testing samples.
- The Glass Identification dataset consists of 9 continuous features of 214 samples. 163 of them are window glass and 51 are non-window glass.
- The Cleveland Heart Disease dataset consists of 303 samples with 13 features. We attempt to distinguish presence of heart disease from absence. There are 139 samples of presence and 164 samples of absence.

In the experiments, the percentage of correctness is evaluated on testing samples. When testing samples are not available, we use the average correctness over 10-folds. Experiment results are summarized in Table 2. For SCAD PSVM, we fix $a = 3.7$ according to [27]. The performance of three penalty functions are similar, and they all achieve noticeable reduction in the dimensionality of input space.

2.5 Conclusions

For hyperplanes refined to certain feature subspace, we derive a bound on the VC dimension that is proportional to the dimensionality of the subspace. This is the motivation of our new approach Penalized Support Vector Machines. An analysis of the dual leads to the penalized least square problem. We discuss the statistical properties following the line of [27]. Our approach trains SVM while doing feature selection automatically and simultaneously.

A main shortcoming of our approach is that one more parameter is included to optimize compared to SVMs. Further research can be continued on how to adjust the GACV method [67] to tune the trade-off parameters.

Table 2: A comparison of SVM and PSVM on six UCI datasets. The results include testing correctness (or 10-fold cross validation correctness) and the average number of features selected.

Dataset	SVM %correctness #features	PSVM (hard) %correctness #features	PSVM (soft) %correctness #features	PSVM (SCAD) %correctness #features
WPBC	75.38 32	75.38 10.8	75.38 2	75.38 4
Ionosphere	87.14 33	88.57 16.4	87.14 20.4	87.14 21
Segmentation	64.95 19	83.24 8	84 10	83.29 10
Pendigits	98.66 16	99.7 12	99.11 12	99.85 12
Glass	92.38 9	92.86 5.2	92.38 5.7	92.38 3.8
Cleveland	80.33 13	83 8.8	80.33 9.4	80.33 9.6

A challenging topic for future research is to study whether the proposed approach perform as well as the oracle procedure for feature selection, a property studied in [27].

CHAPTER III

CASCADE

This chapter is about numerical strategies in searching for an optimal cascade. Section 3.2 describes the general architecture of a cascade model. Section 3.3 describes three types of heuristics, and their related implementation strategies. Section 3.4 describes the experiments and results. Section 3.5 gives some discussions. Some future research topics are raised. Some concluding marks are provided in Section 3.6.

3.1 Introduction

Recently, we have seen a boosting of hierarchical-structure-based methodologies. The following list provides a few examples.

- In statistics, there are CART and MARS [13] and many other follow-up works.
- In machine learning, there are C4.5 [55], perception trees [10], and a lot more...
- In signal processing and harmonic analysis, we have seen quad-tree based image coding [57, 60], pyramid [16], and so on.

A commonality of the above methodologies is that they all rely on a hierarchical structure.

In this chapter, a special type of hierarchical structure is studied. We limit ourselves to a binary classification problem: the response is either 0 or 1. We consider a hierarchy of classifiers such that at each level (in the hierarchy), there are exactly two branches, and at least one of them is terminal: it has no subsequential classifier. For a historic reason, this structure is called a *cascade.*

Cascades are the fundamental structures of many successful techniques in various applications. Some recent works include the Maximum Rejection Classifier attributed to M. Elad et al. [23], a similar method in pattern recognition by H. Hel-Or and Y. Hel-Or [39, 38], a successful framework in computer vision by Viola and Jones [66], and an application in automatic target recognition by a group in M.I.T. [37].

There are many advantages in using a cascade. A few of them will be listed in the following.

1. *Interpretability.* Such a framework can generate a model that is easy to interpret.
2. *Computational efficiency.* In general, it is easy to design a fast algorithm in implementations. The computational complexity of a derived algorithm will be small.

3. *Generality.* It will be easy to incorporate other features, for example, multi-scale features, such as wavelets, curvelets, beamlets, and edgelets.

4. *Flexibility.* The designing of the algorithm is flexible: we can optimize the computing procedure based on prior knowledge of the data.

This chapter is about numerical strategies in searching for an optimal cascade. To study the trade-off between two types of errors (false alarm and mis-detection), a formulation is provided. The basic idea is to derive algorithms that can approximate the frontier of the feasible region of pairs of two types of error rates. This is in spirit similar to the idea of using a Receiver Operating Characteristics (ROC) curve. Three heuristics are presented. They are (1) a propagating method, (2) another approximation method by controlling the error rates, and (3) a method based on minimizing weighted error rates. Simulations are design to empirically examine the properties of each of these methods. It is found that the weighting strategy is optimal in terms of both the error rates and the speed. This finding implies that in designing algorithms for cascade, weighting is more favorable.

We experiment our current cascade to an IR data set. The results do not outperform another existing method: support vector machine (SVM). However, we only allow a very simple type of classifier in our hierarchy; while SVM utilizes nonlinear classifiers. We discuss the limitation of our current framework, and point out the direction for future improvement. Our approach is systematic, generic, and flexible. It has strong promises to be developed into a powerful method. The current suboptimal performance in a particular data set indicates the necessity of such an improvement.

3.2 Architecture of a Cascade

We consider a binary classification problem. A cascade classifier has the following structure:

$$
\begin{array}{l}
\text{Input Data } \mathbf{x} \\
\Downarrow \\
\boxed{f_1(\mathbf{x}) + b_1} \quad \stackrel{<0}{\Longrightarrow} \mathcal{C}_1 \\
\Downarrow \geq 0 \\
\boxed{f_2(\mathbf{x}) + b_2} \quad \stackrel{<0}{\Longrightarrow} \mathcal{C}_1 \\
\Downarrow \geq 0 \\
\boxed{f_3(\mathbf{x}) + b_3} \quad \stackrel{<0}{\Longrightarrow} \mathcal{C}_1 \\
\Downarrow \geq 0 \\
\mathcal{C}_2
\end{array}
\tag{16}
$$

Here $\mathcal{C}_1$ stands for the first class, and $\mathcal{C}_2$ stands for the second class. In our experiment, "$\mathcal{C}_1$" is the clutter class (labelled by "0"s) and "$\mathcal{C}_2$" is the target class (labelled by "1"s).

Each intermediate node is made by a simple classifier:

$$f_j(\mathbf{x}) + b_j \lesseqgtr 0,$$

where j is the index of the intermediate node. The previous diagram gives a cascade classifier with three intermediate nodes. Each $f_j(\mathbf{x})$ may have the form:

1. single entry of $\mathbf{x}$:
$$f_j(\mathbf{x}) = \mathbf{x}_j,$$
where $\mathbf{x}_j$ is the j-th coordinate of vector $\mathbf{x}$;

2. linear:
$$f_j(\mathbf{x}) = c^T\mathbf{x};$$

3. quadratic:
$$f_j(\mathbf{x}) = c^T\mathbf{x} + \mathbf{x}^T M\mathbf{x}; \qquad \text{or}$$

4. support vector machine (SVM) with kernels which are either higher than degree-2 polynomials, or radial basis functions
$$f_j(\mathbf{x}) = \sum_{k=1}^{N} \alpha_k K(\mathbf{x}, \mathbf{x}_k),$$
where $K(\cdot,\cdot)$ is a kernel function, $\mathbf{x}_k$'s are the N observations and α_k's are the weights that are trained by applying a SVM.

In our simple classifier, an observation is predicted to be in class $\mathcal{C}_2$ if and only if

$$f_j(\mathbf{x}) + b_j \geq 0, \text{ for all } j;$$

otherwise the prediction is $\mathcal{C}_1$. We choose b_j small enough such that the chance that an observation from $\mathcal{C}_2$ is misclassified as an observation from $\mathcal{C}_1$ is small. When a cascade is built, the thresholds b_j's will remain unchanged afterwards.

Note that at the early stage of a cascade algorithm, we would like to choose a simple kernel. In later stages of a cascade algorithm, we tend to choose more complex kernels, to explore more difficult structures.

In building a cascade model, in the second node, we exclude all the training data that are predicted by the first classifier as in $\mathcal{C}_1$. In other words, the second simple classifier is built for the "remaining" observations that are classified as in $\mathcal{C}_2$ by the first classifier. In general, the consequent classifier is built for the data that are classified as $\mathcal{C}_2$ by the

previous classifier. Apparently the number of observations decreases, therefore the training of classifiers becomes easier.

For simple decision rules, we can apply Linear Discriminant Analysis (LDA). When a more complex model is needed, we can use Quadratic Discriminant Analysis (QDA). A description on how to find a decision rule via LDA and QDA can be found in [34]. When a more complex decision rule is needed, support vector machines can be applied, for example, with a radial basis function as kernel. More details on how to train support vector machines can be found in [18].

We imagine the following simple procedure. A cascade starts with simple linear detectors (function f_i's). If targets are contained in a subregion of the space that is made by all image chips, a sequence of linear detection rules will outline a convex hull of the subregion that include (hopefully) most of the targets. See Figure 2 for an illustration.

There is a point in which no linear classifier will be effective in distinguishing a significant proportion of clutters from targets. However, it can still be true that by utilizing a more complex rule, e.g., quadratic or radial basis SVMs, it is possible to distinguish a significant amount of clutters from targets. A cascade approach shall automatically switch to a more complex decision rule when it is necessary. Figure 3 shows that when the target region has holes, support vector machines with radial basis functions can render an efficient classifier, while both LDA and QDA will fail. Note that since many clutters have been "rejected" in the previous stages, the sample size at the later stage is small. Hence support vector machines can be trained efficiently.

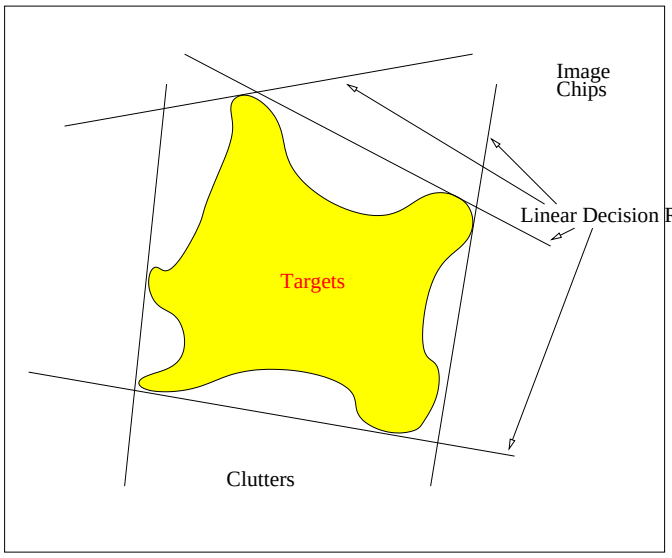


Figure 2: A set of linear classifiers can quickly find the convex hull of a target region.

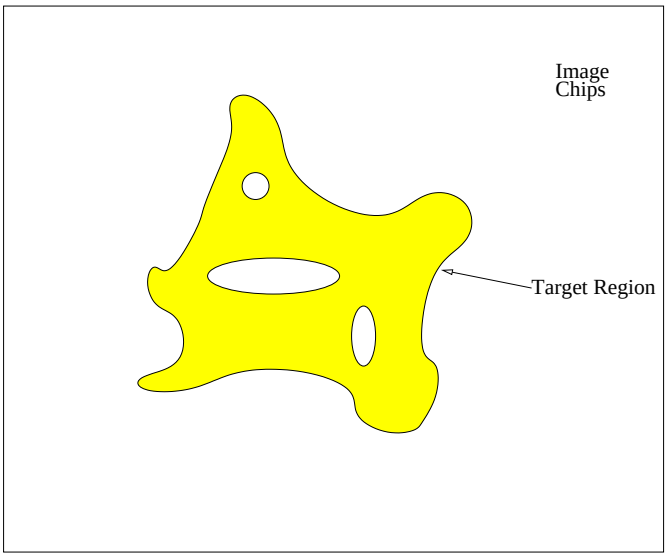


Figure 3: A target region with 'holes'. A radial basis SVM will find a good classifier in such a situation, while a set of linear classifiers will not.

3.3 The Optimal Strategies in Building a Cascade

We consider what the optimal strategies are in building a cascade classifier. We limit our analysis on binary classification problems. The general description of a cascade is given. The optimality of a cascade is defined in an empirical sense. Finally, we propose some numerical experiments to compare different strategies.

Section 3.3.1 formulates the problem. It provides the necessary notations to analyze the model. Section 3.3.2 describes three types of heuristics, and strategies of implementations under various circumstances.

3.3.1 Cascade Classifier

Consider the diagram (16). For a fixed natural number L, a cascade with L levels can be written as

$$y = \begin{cases} 1, & \text{if } f_1(\mathbf{x}) \geq t_1, \quad f_2(\mathbf{x}) \geq t_2, \quad f_3(\mathbf{x}) \geq t_3, \ldots, \quad f_L(\mathbf{x}) \geq t_L; \\ 0, & \text{otherwise,} \end{cases}$$

where symbol $\mathbf{x}$ denotes the input, and symbol y denotes the response, which is binary (0 or 1). The functions $f_i(\cdot), i = 1, 2, \ldots, L$, are relatively simple, e.g. linear functions, univariate functions, etc. The quantities $t_i, i = 1, 2, \ldots, L$, are constants. For example in CART [13], a simple classifier has the form:

$$\mathbf{x}_j \geq c, \tag{17}$$

where $\mathbf{x}_j$ is the jth component of a random vector $\mathbf{x}$, and c is a constant. Let $f_1(\mathbf{x}) = \mathbf{x}_j$ and $t_1 = c$, then the classifier $f_1(\mathbf{x}) \geq t_1$ is equivalent to the classifier in (17).

To evaluate the performance of a given cascade, we consider an empirical approach.

Let N denote the total number of observations. Let $N^i, i = 0, 1$ denote the numbers of observations that belong to classes '0' and '1' respectively. Let N_1 denote the number of observations that are classified as in class '0' by decision rule "$f_1(x) < t_1$". Let N_2 denote the number of remaining observations. In general, among the N_{2i-2} observations that are classified as '1' in the $(i-1)$th step, let N_{2i-1} denote the number of observations that are classified as class '0' by decision rule "$f_i(x) < t_i$", $1 \le i \le L$; and let N_{2i} denote the number of remaining observations. Due to the hierarchical structure of a cascade, we must have

$$\begin{aligned} N &= N_1 + N_2, \\ N_{2i-2} &= N_{2i-1} + N_{2i}, \quad i = 2, 3, \ldots, L, \text{ and} \\ N^j_{2i-2} &= N^j_{2i-1} + N^j_{2i}, \quad i = 2, 3, \ldots, L, j = 0, 1. \end{aligned}$$

Among the $N_j, 1 \le j \le 2L$, observations, let $N^i_j, i = 0, 1$, denote the number of observations that belong to class '0' and '1' respectively. The number of 1's which are misclassified as 0's is

$$\sum_{i=1}^{L} N^1_{2i-1}.$$

At the same time, the number of 0's which are misclassified as 1's is

$$N^0_{2L}.$$

The above quantities are embedded in a cascade according to the following diagram:

$$\left.\begin{array}{c} N = N^0 + N^1 \\ \downarrow \quad \searrow \\ N^0_2 + N^1_2 = N_2, \quad N_1 = N^0_1 + N^1_1 \\ \downarrow \quad \searrow \\ N^0_4 + N^1_4 = N_4, \quad N_3 = N^0_3 + N^1_3 \\ \downarrow \quad \searrow \\ N^0_6 + N^1_6 = N_6, \quad N_5 = N^0_5 + N^1_5 \\ \downarrow \quad \searrow \\ \vdots \\ \downarrow \quad \searrow \\ N^0_{2L} + N^1_{2L} = N_{2L}, \quad N_{2L-1} = N^0_{2L-1} + N^1_{2L-1} \\ \downarrow \\ \text{'1'} \end{array}\right\} \rightarrow \text{'0'} \tag{18}$$

A hierarchical classifier that has the above architecture is called a *level-L cascade.* Recall that each simple classifier is a binary classifier. Let $\mathcal{H}(L)$ denote all the level-L cascade. For a given level-L classifier h (i.e., $h \in \mathcal{H}(L)$), let $E_{0\to 1}(h)$ denote the number of 0's that are

wrongly classified by h as 1's; and let $E_{1\to 0}(h)$ denote the number of 1's that are wrongly classified as 0's. If h produces the results in the diagram (18), we have

$$E_{1\to 0}(h) = \sum_{i=1}^{L} N_{2i-1}^{1}, \quad E_{0\to 1}(h) = N_{2L}^{0}.$$

We define the following function:

$$\begin{aligned} f(k;L) = \quad & \min_{h\in\mathcal{H}(L)} \quad E_{0\to 1}(h), \\ & \text{subject to} \quad E_{1\to 0}(h) \le k, \end{aligned}$$

where k is a positive integer. A cascade h is optimal if and only if the following condition is satisfied:

$$E_{0\to 1}(h) = f[E_{1\to 0}(h); L].$$

Or in other words, the optimal cascade should reside on the *frontier* of the feasible region. An illustration is shown in Figure 4.

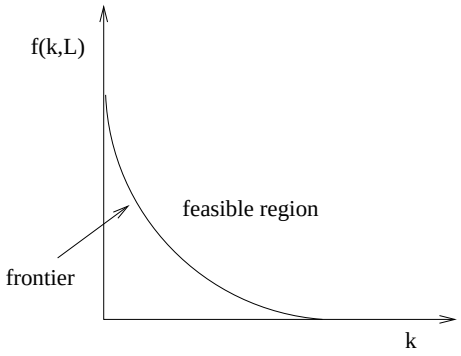


Figure 4: An illustration of the frontier

We consider strategies that can compute an optimal cascade. There are three classes of heuristics:

1. Frontier-following (in Section 3.3.2.1),
2. Controlled error rates (in Section 3.3.2.2), and
3. Weighting (in Section 3.3.2.3).

Before describing the above heuristics in details, we would like to make the following general comments:

1. We conjecture that finding the optimal cascade in a generic situation is a hard problem, especially when the number of levels L is large.

2. The current formulation does not address the generalization error rate. To study the generalization error, we may take advantage of the knowledge in *structured risk minimization* [64].

3. Due to the difficulty of computing the optimal cascade, we hope to utilize some greedy algorithm to find a suboptimal cascade. The heuristics, which will be described later, are ways to explore different possibilities in this problem.

4. The *frontier* gives a set of cascade classifiers. The particular choice of a cascade depends on the cost of different types of errors. It is problem dependent, and we choose not to pursue in this direction.

3.3.2 Three Heuristics

We describe three types of heuristics. Section 3.3.2.1 describes strategies which follow the frontier of the feasible region. Section 3.3.2.2 describe a heuristic that is based on controlled error rates. Section 3.3.2.3 describes a strategy that is based on the weighted misclassification rate.

3.3.2.1 Frontier Following

Optimal Cascade. Let $h^*(k, L)$ denote the optimal cascade classifier with L levels and k observations which are wrongly classified as 0's. We have

1. $E_{1\to 0}([h^*(k, L)] = k$, and

2. $E_{0\to 1}[h^*(k, L)] = f(k; L)$.

A single cascade classifier, $h^*(k, L)$, corresponds to a point on the frontier.

Computing $h^*(k, 1)$. When the level of a cascade classifier is 1, in many cases, there are fast algorithms to compute the classifier $h^*(k, 1), k = 0, 1, 2, \ldots, N^1$. As an example, if the classifier are those that are used in CART:

$$\mathbf{x}_j \geq a,$$

For each variate $\mathbf{x}_j$, under the condition $E_{1\to 0} \leq k$, one can find the corresponding maximal error rate for $E_{0\to 1}$. Note that the above problem is for a univariate distribution. The $h^*(k, 1)$ is obtained by taking the maximum for all $\mathbf{x}_j$'s.

Propagating. We develop a propagation algorithm to *approximate* the frontier. We start with some notations.

For a cascade, let $R^1(h)$ denote the region in which the response is predicted to be 1:

$$R^1(h) = \{x : h(x) = 1\}.$$

Let H_{A,k_1} denote all the simple classifiers in region A, whose error rate $E_{1\to 0}$ are no larger than k_1. Apparently, if a classifier $h \in H_{A,k_1}$, we have $E_{1\to 0}(h) \le k_1$. Note that the domain of the classifier h is the region A. Let $r^*(A, k_1)$ denote the classifier which is in the set H_{A,k_1} and takes the minimum of $E_{0\to 1}$:

$$r^*(A, k_1) = \arg \min_{h \in H_{A,k_1}} E_{0\to 1}(h).$$

We may use the following heuristic to approximate the frontier. Note that we can not guarantee that the exact frontier will be found:

Algorithm **Prop**.

1. Let $h'(k, 1) = h^*(k, 1), k = 0, 1, 2, \ldots, N^1$.

2. **For** $\ell \ge 1$,
   let $h'(k, \ell+1), k = 0, 1, 2, \ldots$ denote the solution to the following optimization problem:

$$\min_{k_1 \le k} \quad E_{0\to 1}[h'(k_1, \ell)] + E_{0\to 1}\big(r^*\{R^1[h'(k_1, \ell)], k - k_1\}\big), \tag{19}$$

   End.

3. Use $h'(k, L)$ to approximate $h^*(k, L), 1 \le k \le N^1$.

Note that the number N^1 in the first row of the above algorithm can be reduced to a small value, given that the error rate, $E_{1\to 0}$, can be controlled significantly smaller than the quantity N^1. By doing so, a large amount of computation can be saved.

An Implementational Strategy. We describe a numerically appealing alternative to implement the algorithm in Section 3.3.2.1. Let K denote a fixed integer. We consider how to compute $h'(k, \ell + 1), 0 \le k \le K$, while $h'(k, \ell)$'s are given. The main idea can be illustrated by the following table:

	0	1	2	$\cdots$	$K-1$	K
	$h'(0,\ell)$	$h'(1,\ell)$	$h'(2,\ell)$	$\cdots$	$h'(K-1,\ell)$	$h'(K,\ell)$
	A_0	A_1	A_2	$\cdots$	A_{K-1}	A_K
0	$r^*(A_0,0)$	$r^*(A_1,0)$	$r^*(A_2,0)$	$\cdots$	$r^*(A_{K-1},0)$	$r^*(A_K,0)$
1	$r^*(A_0,1)$	$r^*(A_1,1)$	$r^*(A_2,1)$	$\cdots$	$r^*(A_{K-1},1)$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$			
$K-2$	$r^*(A_0,K-2)$	$r^*(A_1,K-2)$	$r^*(A_2,K-2)$			
$K-1$	$r^*(A_0,K-1)$	$r^*(A_1,K-1)$				
K	$r^*(A_0,K)$					

(20)

In the above table, we have

$$A_i = R^1[h'(i,\ell)], \quad i = 0, 1, \ldots, K.$$

Following a similar description as in Section 3.3.2.1, for entries in one column of the above table,

$$r^*(A_i, k), k = 0, 1, \ldots, K - i,$$

there are efficient algorithms to compute. The solutions to the problem in (19) can be quickly extracted by comparing the values of the objective function in (19) at entries that are in a line which is parallel to the second diagonal, i.e. entries $(0, i), (1, i-1), \ldots, (i-1, 1), (i, 0)$. In implementations, this approach should give an efficient algorithm.

One question is how to integrate the above algorithm with specific types of classifiers, e.g. LDA or QDA. We provide a few suggestions. In the LDA, the discriminant direction can be computed first. Then the threshold is varied to achieve different error rates. In the QDA, similar approach can be taken: finding the quadratic form first, then the threshold is varied to obtained different error rates.

The complexity of a propagating algorithm can be relatively high, especially when the error rate $E_{1\to 0}$ is large. Suppose the simple classifiers are based on single variate functions, as in (17). For a fixed coordinate, the computational complexity of finding the optimal classifier should be at least the order of complexity to sort the N coordinates, which is $O(N \log(N))$. Let d denote the dimensionality of the data: there are d variates. The propagating algorithm searches in table in (20), which has $O(K^2)$ entries. Overall, the computational complexity is $O(K^2 \cdot d \cdot N \log(N))$.

3.3.2.2 Controlled Error Rate

Sometimes we may want to quickly compute a sub-optimal cascade. The following approach can be adapted.

Algorithm **CER**.

1. Let $\tilde{h}_0$ be a classifier, which classifies everything to class '1':

$$\tilde{h}_0(x) = 1, \forall x.$$

2. Set $k = 0$, $\ell = 0$.

3. **While** the last improvement is greater than 0, (in the first time, this condition is always true,)

   Find $r^*[R^1(\tilde{h}_\ell), 0]$;

   Let $\tilde{h}_{\ell+1}$ be the combination of $\tilde{h}_\ell$ and $r^*[R^1(\tilde{h}_\ell), 0]$;

Update: $\ell \leftarrow \ell + 1$ and $\tilde{h}_\ell \leftarrow \tilde{h}_{\ell+1}$.
Record the last improvement as $E_{0\to1}(\tilde{h}_{\ell+1}) - E_{0\to1}(\tilde{h}_\ell)$;

End.

4. Set $h'(k) = \tilde{h}_\ell$.

5. Set $k \leftarrow k + 1$.
Find $r^*[R^1(\tilde{h}_\ell), 1]$;
Let $\tilde{h}_{\ell+1}$ be the combination of $\tilde{h}_\ell$ and $r^*[R^1(\tilde{h}_\ell), 1]$;
Update: $\ell \leftarrow \ell + 1$ and $\tilde{h}_\ell \leftarrow \tilde{h}_{\ell+1}$.

6. If $k = K$, where K is a predetermined maximal allowable error rate, $E_{1\to0}$, then stop;
Otherwise, go back to the step 4.

7. The function $h'(k), k = 0, 1, 2, \ldots, K$, give us an approximation to the frontier.

Let $f(k)$ denote the lower bound of all cascades (the number of levels can be arbitrarily large):

$$f(k) = \min_L f(k; L).$$

Let $h^*(k)$ denote the classifier which corresponds to $f(k)$, or in other words,

$$h^*(k) = \text{argmin}_{h^*(k,L), L=1,2,3,\ldots} E_{0\to1}[h^*(k, L)].$$

The intuition of designing the above procedure is that hopefully, the final $\tilde{h}_\ell$ is close to the optimal classifier, $h^*(k)$.

We expect this approach to be very computationally efficient. This algorithm is analogous to the ideas in the MRC.

3.3.2.3 Weighting

We propose an alternative to the method *controlled error rate*. This method is also motivated by using Lagrangian multipliers to locate the frontier of a feasible region.

Let λ be a potentially large positive constant. By following a similar argument in Section 3.3.2.1, the following optimization problem can be solved efficiently.

$$s^*(A, \lambda) = \text{argmin}_{h\in\mathcal{H}(A)} \quad E_{0\to1}(h) + \lambda E_{1\to0}(h), \tag{21}$$

where A denotes a region, the set $\mathcal{H}(A)$) includes all simple classifiers residing in the region A, and λ is a positive constant.

The starting value of λ is typically large. One can gradually reduce the value of λ to produce a classifier that minimizes both $E_{0\to1}$ and $E_{1\to0}$ simultaneously.

We propose the following algorithm.

Algorithm **Weighting**.

1. Let $\tilde{h}_0$ be a classifier, which classifies everything to class '1':

$$\tilde{h}_0(x) = 1, \forall x.$$

2. Set $\ell = 0$.

3. Choose a large initial value (denoted by λ_0) for λ: $\lambda = \lambda_0$.

4. **While** the last improvement in the objective function in (21) is 'significantly nonzero',
   find $s^*[R^1(\tilde{h}_\ell), \lambda]$;
   Let $\tilde{h}_{\ell+1}$ be the combination of $\tilde{h}_\ell$ and $s^*[R^1(\tilde{h}_\ell), \lambda]$;
   Update: $\ell \leftarrow \ell + 1$ and $\tilde{h}_\ell \leftarrow \tilde{h}_{\ell+1}$;
   Record the decreasing of the objective function that is in (21).
   The above is the "last improvement".
   End.

5. Let $k = E_{1\to 0}(\tilde{h}_\ell)$, $f(k) = E_{0\to 1}(\tilde{h}_\ell)$.

6. Let λ take a smaller value: $\lambda \leftarrow \lambda/\theta$, e.g. $\theta = 2$.

7. If $f(k) < 1$, then go back to the step 4; Otherwise, continue.

8. Output $[k, f(k)]$ to approximate the frontier.

Note that by controlling the final value of parameter λ, we can realize different trade-off between the two error rates: $E_{0\to 1}$ and $E_{1\to 0}$.

3.4 Experiments

In this section, our objective is to assess the performance of the above three heuristic algorithms. Experiments were carried out on two synthetic data sets and an infra-red (IR) image data. For simplicity, we stayed with univariate functions employed in CART. In the results, the approximate frontier f-curves reflected the empirical performance on training data. And the generalization performance on testing data was evaluated by the complementary ROC (c-ROC) curves, with x-axis representing the missed detection rate and y-axis representing the false alarm rate. Each point on an f curve corresponded to a cascade, which was used to generate a point on the c-ROC curve.

All experiments were conducted in MATLAB codes.

3.4.1 Synthetic Data

In the following two examples, we simulated two data sets with different underlying distributions and compared the performance of three heuristic algorithms. For the method *propagating*, the levels of cascades were assigned as $L = 2n$, where n is the number of dimensions. For *weighting*, We set $\lambda = 40$ and $\theta = \sqrt{2}$.

Example 1: $X \in \mathbb{R}^2$. The probability of $y = 1$ is 0.3, and $y = 0$ is 0.7. If $y = 1$, X are drawn from $N(\mathbf{0}, I)$, otherwise X are drawn from $N(\mathbf{0}, 10I)$. The theoretical boundary is

$$x_1^2 + x_2^2 = \frac{20}{9}\log(\frac{30}{7}).$$

An explanation on the above decision rule is postponed to Appendix 3.7.

Example 2: $X \in [0,1]^{10}$. The two classes have equal prior probability. Define the region $A = \{X : 0.3 \le X_1 \le 0.7, 0.4 \le X_2 \le 0.6, 0.2 \le X_3 \le 0.4\}$, and denote $|A|$ the area of region A. The conditional density of X in class i $(i = 0, 1)$ is as follows

$$f_i(X) = \begin{cases} \alpha_i, & X \notin A, \\ \frac{1}{|A|}(1 - \alpha_i) + \alpha_i, & X \in A. \end{cases}$$

In the first example, we generated 1000 observations for training and 1000 for testing. In the second example, we assign $\alpha_1 = 0.2$ and $\alpha_2 = 0.8$, and both training and testing data sets contain 1000 observations for each class. Simulated data sets for these two examples are shown in Figure 5. The running time comparisons of three algorithms are given in Table 3. Figure 6(a) and 7(a) contain the f-curves based on training data, and Figure 6(b) and 7(b) are the c-ROC curves for testing data. For both examples, *weighting* and *propagating* approximate the theoretical boundaries very well, which implies their good generalization property. A small zigzag phenomenon was observed on the c-ROC curves generated by *propagating*. In Figure 8, we study the phenomenon by connecting all pairs of corresponding points, and it is clearly seen that the whole trend is homogeneous despite some small random movements.

3.4.2 An IR Data

The method *weighting* seems most promising when taking both the performance and running time into consideration. We applied *weighting* to an infra-red (IR) image data set. The data set consists of 20000 training examples and 6898 testing examples of 300 dimensions. The results are shown in Figure 9. In the c-ROC curve plot, we compared its performance with Maximal Rejection Classifier (MRC). It is clear that our method outperforms MRC.

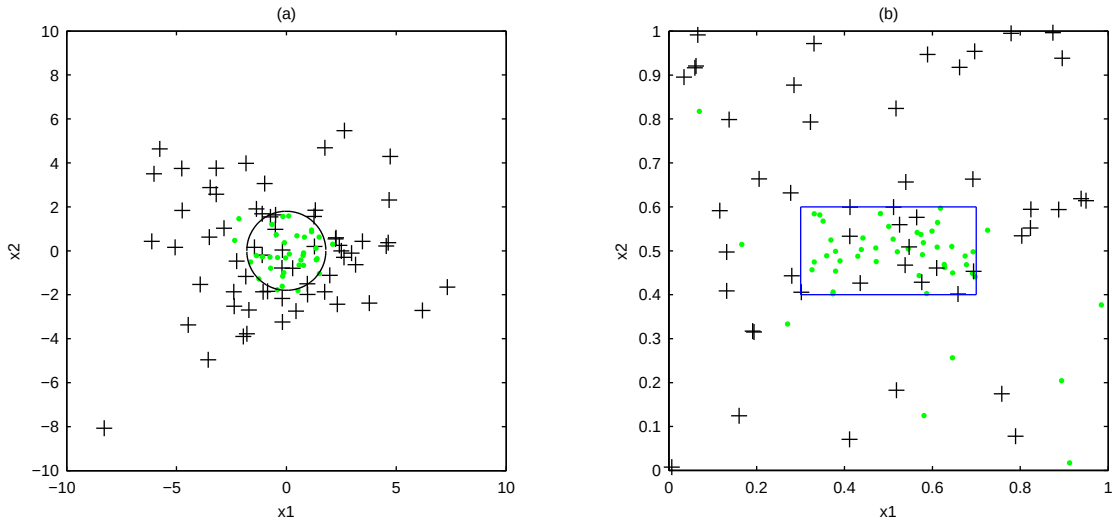


Figure 5: Illustrations of two artificial data sets with theoretical boundaries. (a) Example 1: two Gaussian distributions, with different variances, (b) Example 2: two mixed Uniform distributions, viewed from the first two dimensions, with different mixing parameters.

3.5 Discussion

We will discuss the possibilities of techniques that can be integrated into the above framework. We also discuss the relation of our approach to other existing methodologies.

3.5.1 Related works

Connection with MART and Boosting. MART [31, 30] is essentially a boosting method for tree models. Boosting has been proven to be an effective method in training classifiers, see [32, 58]. A disadvantage of both MART and a direct output of boosting is that the trained classifier can be very complex: it is an additive model having a large number of components. Comparing to MART, a hierarchical model is much easier to implement, and will be guaranteed to be fast. The training of a hierarchical model can be faster than boosting. However, we expect boosting to provide a better performance.

It is interesting to note that in a successful application in computer vision, cf. [66], the researchers first use a boosting approach to find an 'ideal' model. They then use a cascade (hierarchical model) to 'approximate' the 'ideal' one. They reported superior performance in simulations.

Connection with MRC. MRC [23] and an enhanced approach [39, 38] use the structure of a cascade detector. The difference from our proposed method is that in training an intermediate detector f_i, we will allow different types of classifiers, e.g. SVM. The function f_i can be highly nonlinear. This has not been systematically studied in existing framework. Moreover, we will research on how to choose the function type for f_i *adaptively*.

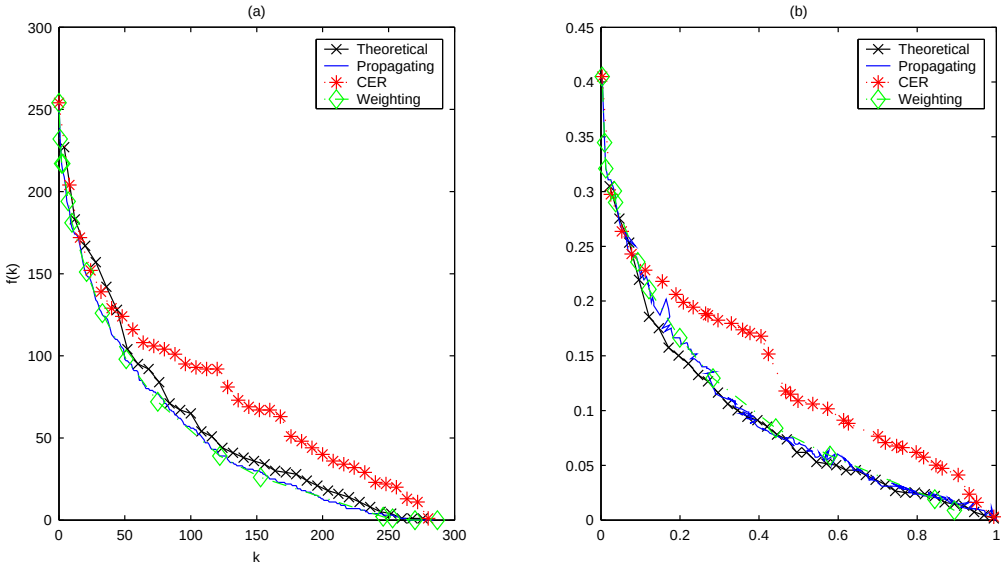


Figure 6: A comparison of three algorithms for Example 1. (a) f-curves for training data, (b) c-ROC curves for testing data. It shows that *CER* always is the worst, and *Weighting* is nearly as good as the *Propagating*. For the testing data, the closeness of *Weighting* and *Propagating* implies a good generalization properties of them in this setting.

Conceptually, MRC is similar to *CER*.

3.5.2 Theoretical questions

The following are some theoretical problems for constructing a cascade.

1. What is the optimal strategy in deciding the single classifier at each step of a cascade?
2. Given a cascade, how to evaluate its rate of convergence? And in which sense?
3. How to characterize the generalization error of a cascade?
4. How to characterize the *rate of approximation* of a searching method to the frontier of (a class of) cascades.
5. How to guarantee the consistency of a computed cascade?

The detailed discussion on the above problems will be postponed for future publications.

3.5.3 Nonparametric approaches

Zhu and Hastie [72] recently proposed a nonparametric method in classification. They consider the likelihood ratio between two classes. The likelihood ratio is estimated by some kernel based nonparametric density estimation methods. In their framework, LDA and QDA become special cases. They provide evidences that their method can successfully classify

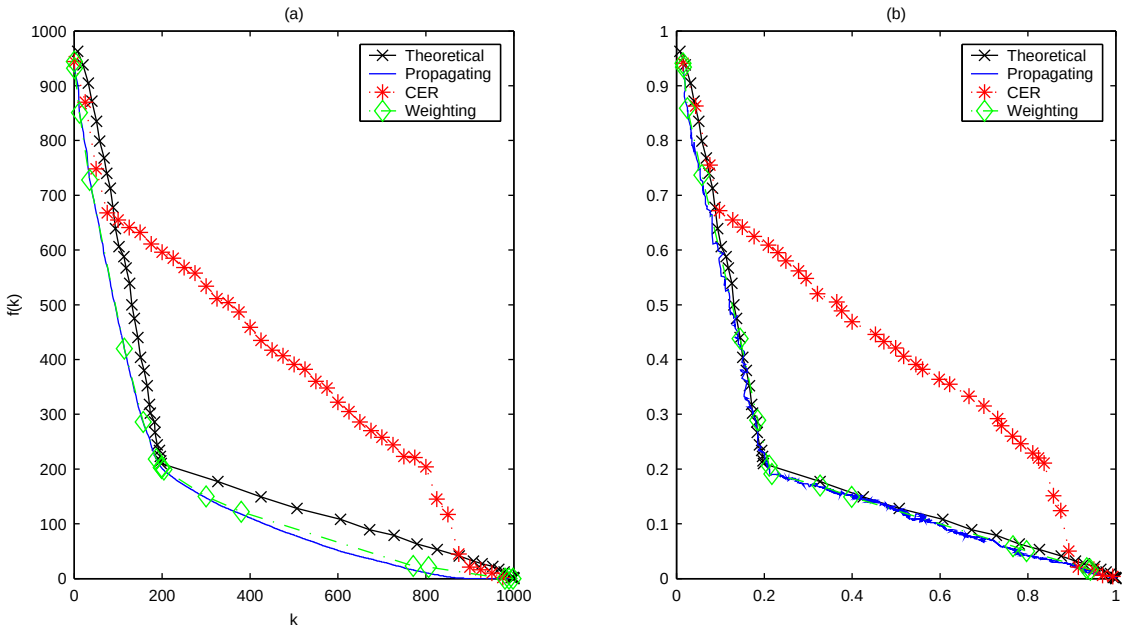


Figure 7: A comparison of three algorithms for Example 2. (a) f-curves for training data, (b) c-ROC curves for testing data. Similar conclusions as in Example 1 can be drawn here.

data, in the situations that are hard for LDA and QDA. Their work can be considered as a generalization to the existing paradigm of LDA and QDA. It will be interesting to examine how to integrate their method into a cascade algorithm.

3.5.4 More on decision rules that are based on marginal distributions

Many questions that we asked above eventually transfer to how to design a classifier in a univariate case. In fact, many of the classifiers are combinations of a projection followed by a univariate classifier. If we can understand better about what an optimal univariate classifier should be, we have a good guidance to design a cascade. Note that so far, the decision rule

$$f_i(\mathbf{x}) \lessgtr 0$$

implies a simple cut-off decision rule. In the sense of Neyman-Pearson, such a classifier is only optimal when the likelihood ratio is a monotonic function of the variable. We will study the case with more general assumptions and more data-driven methods.

3.5.5 Regularization

Another interesting problem is to study whether the regularization (or the penalized likelihood methods) can be a framework to build a cascade model. If yes, what is(are) the advantage(s)?

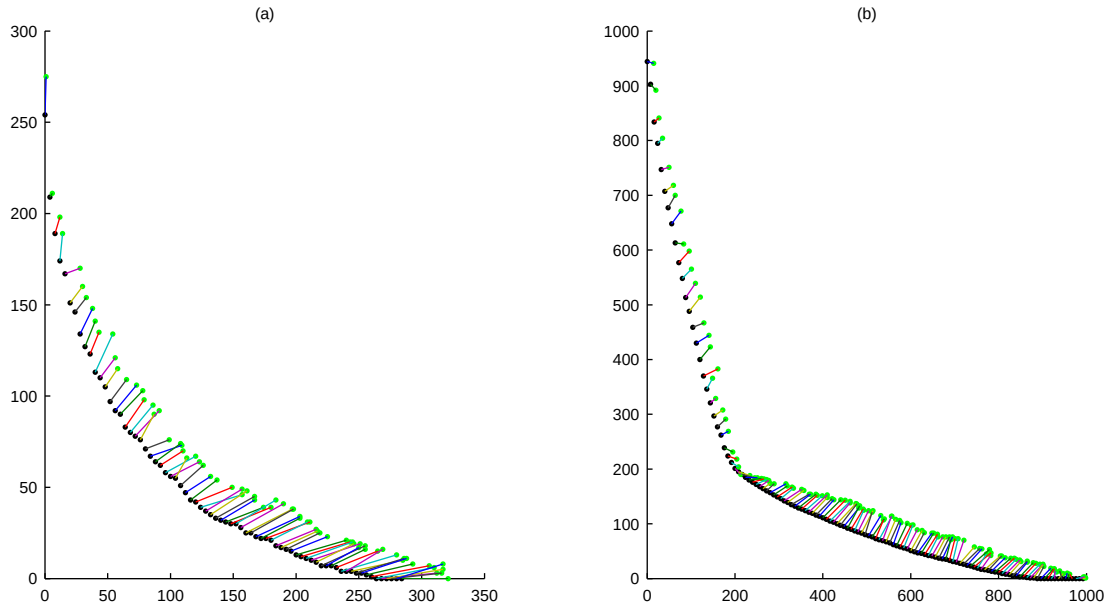


Figure 8: Dark dots denote the points on f-curve via *propagating* on the training data and light dots denote the points on the c-ROC curve for the same classifiers using the testing data. A line is drawn to connect a corresponding pair. Figure (a) is for Example 1, and (b) for Example 2. These figures show how the error rates change while the trained classifiers, which are trained by the *training* data, are applied to the *testing* data.

3.5.6 How to integrate multi-scale features

An advantage of a cascade approach is its flexibility to incorporate various types of features, e.g. multiscale features.

As a matter of fact, if we choose LDA or QDA to train the detector f_i as in Section 3.2, the resulting linear or quadratic decision rule gives us a 'feature'. Since multiscale analysis has provided a set of feature, we should take advantage of them. Let $\mathbf{z} = \{z_1, z_2, \ldots, z_T\}$ be the set of features. (Note that each z_i can be a linear transform of the observation $\mathbf{x}$.) An intuitive idea is to restrict a function f_i to be a univariate function of those features z_i's. By doing so, we avoid re-training from the observation $\mathbf{x}$. To efficiently determine function f_i, one can adopt the algorithm that has been used in CART [13]. In fact, for our purpose, a simplified version will suffice.

Note that in many multiscale transforms, e.g. wavelets, beamlets, wedgelets, and ridgelets, there exist fast discrete transform. Since the searching of an optimal univariate function f_i is fast too. There will be an fast implementation for the entire procedure.

It is possible to consider f_i as a function of a small number of features z_j's. By doing so, we can incorporate the interaction between features. Note that function f_i can be highly nonlinear, and nonseparable.

Table 3: Training times (in seconds) comparison for three algorithms.

	Propagating	CER	Weighting
Example 1	40.8	0.3	1.2
Example 2	3861.8	6.7	21.2

3.5.7 The necessity of using more flexible classifiers

We compare the results between a cascade (using simple classifiers as "$\mathbf{x}_j > c$") and support vector machine using radial basis functions. Both methods are applied to an IR data set. The results are plotted in Figure 10. It is found that SVM outperforms a cascade. It is mentioned earlier that a set of linear classifier can only detect the convex hull of a target region. The experimental results seem to indicate that the target region is *not* convex. By using more complex simple classifiers in a cascade, this under-performance is likely to be overcome. We leave this as a future work.

3.6 Conclusion

We studied three heuristics in building a cascade. The three heuristics are (1) frontier following approximation, (2) controlling error rates, and (3) weighting. Simulations on synthetic data are carried out. It is found that weighting heuristic is optimal in both computational complexity and error rates. The current winner – weighting algorithms – is applied to an IR data set. It is found that it outperforms some state-of-the-art approaches that utilize the same type of simple classifiers. However, it fails to some classifiers that utilize more complex decision rules.

The contribution of this chapter is to initiate a systematic comparison on several potential heuristics that can be utilized in building a hierarchical model. We point out the implications and promises of the cascade architecture. We describe the feasibilities of integrating a range of techniques into this framework. Many future research directions are discussed.

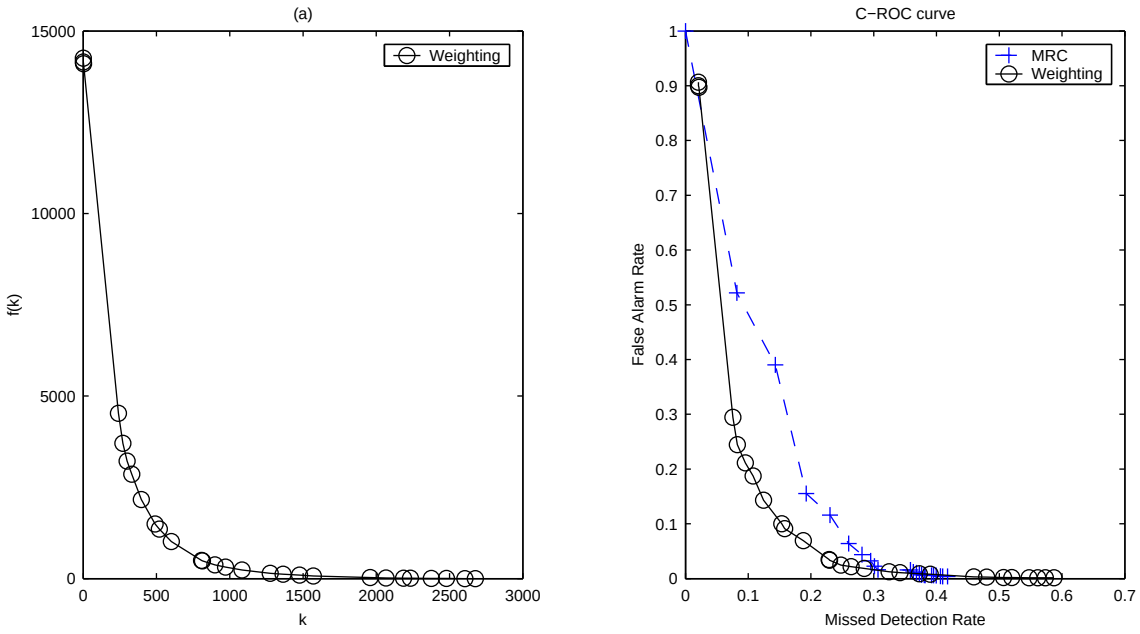


Figure 9: IR image data. (a) f-curve, (b) c-ROC curve: a comparison with MRC.

3.7 Appendix: Theoretical boundary for Example 1

The class conditional probability has Gaussian distribution

$$f_i(x) = \frac{1}{(2\pi|\Sigma_i|)^{1/2}} e^{-\frac{1}{2}x^T\Sigma_i^{-1}x}, \ i = 0, 1.$$

The posterior distribution

$$Pr(Y = i|X = x) = \frac{f_i(x)Pr(Y = i)}{\sum_{i=0,1} f_i(x)Pr(Y = i)}, \ i = 0, 1.$$

In our example, we have $\Sigma_0 = 10I$, $\Sigma_1 = I$, and $Pr(Y = 0) = 1 - Pr(Y = 1) = 0.7$. The log-ratio is

$$\begin{aligned} \log \frac{Pr(Y = 1|X = x)}{Pr(Y = 0|X = x)} &= \log \frac{f_1(x)}{f_0(x)} + \log \frac{P(Y = 1)}{P(Y = 0)} \\ &= -\frac{1}{2}x^T\Sigma_1^{-1}x + \frac{1}{2}x^T\Sigma_0^{-1}x + \log \frac{|\Sigma_0|^{1/2}}{|\Sigma_1|^{1/2}} + \log \frac{3}{7} \\ &= -\frac{9}{20}(x_1^2 + x_2^2) + \log \frac{30}{7} \lessgtr 0. \end{aligned}$$

So a decision boundary can be described as

$$(x_1^2 + x_2^2) \lessgtr \frac{20}{9} \log \frac{30}{7}. \tag{22}$$

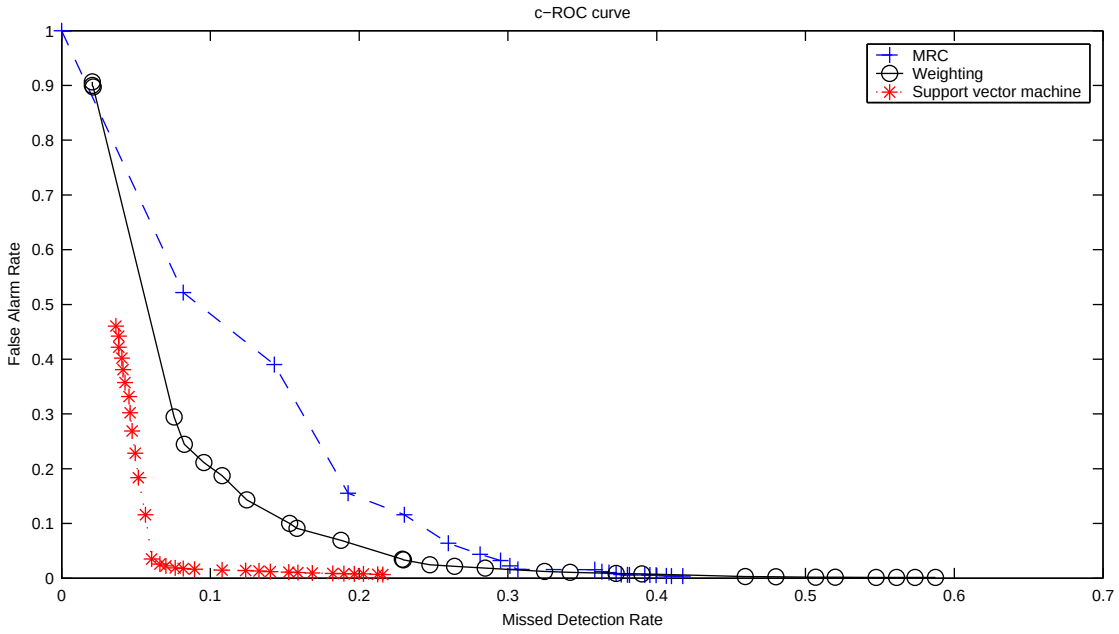


Figure 10: The comparison among MRC, Weighting and SVM on an IR data. In this case, SVM renders the best performance. The Weighting use a single-coordinate-based classifier: $\mathbf{x}_j > a$. This demonstrates the importance of using a more complex classifier at each stage.

CHAPTER IV

BEAMLET CODING

In this chapter, a multiscale coder for curves and boundaries is presented. The chapter is organized as follows. In Section 4.2, beamlets and how to code a single beamlet are described. Section 4.3 describes beamlet representations, as well as how to compute a Rate-distortion optimized beamlet representation. The coding scheme is presented in Section 4.4. Some simulation results are presented in Section 5.3.4. Section 4.6 presents some discussion. In Section 4.7, some concluding marks are provided.

4.1 Introduction

Compressing and coding linear features has a long standing history. They are usually under the topics of *contour coding* and *shape coding*. Shape coding has many important applications in modern multimedia signal processing. It has been used in content based video coding. Editorial [26] and papers included in that special issue provides a full spectrum of related information. An efficient method to code the boundary (i.e., shape) of an object may subsequently facilitate efficient image or video coding.

On the other hand, some artificial images – such as line drawing, maps, cartoons, and blue prints – are mainly composed by lines and curves. A coding scheme that is particularly suitable for linear or curvilinear features is expected to work better in compressing these images.

Our method is a *curve coding* method. It is different from shape coding. Because unlike the boundary of an object (which is called a shape), a curve does not need to be closed. Moreover, curves may have intersections. Examples of images that our method can deal with can be found in Section 5.3.4 Simulations, e.g. Figure 21.

Curve coding has a long standing history. Back in 60's, runlength coding first emerged [28]. The basic idea of a runlength coder is to treat a binary image as a nearest neighbor graph – a pixel is connected to its four (resp. eight) nearest neighbors – and one codes the relative positions of the neighboring pixels. Soon after, a chain coding method is proposed [29]. The key idea of a chain coder is to code, at each step, multiple and aligned pixels (that form a line segment) instead of one nearest neighbor. Many improvements have been introduced for chain coding. For example, in [49], statistical modeling and arithmetic coding is added as a post-processing tool to fully take advantage of the statistical structure in a chain of line segments. Alternatives other than straight line segment have been explored as basic elements. These works include discrete arcs [11], polygonal curves [46], and a lot

more. Researchers have noticed that it is not necessary to exactly follow a curve given by a discrete image. Methods taking into account of the trade-off between the fidelity and efficiency of a contour codec are studied in [59]. More interestingly, the idea of using multiscale structure (i.e. pyramid) to code a contour has been studied by several researchers, e.g. [48, 50]. Despite all the existing work, the authors have not seen a work that applies zero-tree coding for curves. Moreover, the usage of Beamlets, or any structure alike, has not been addressed explicitly in the literature. Part of the intent of this paper is to develop a fully multiscale coding method for curves. The idea of zero-tree coding, which is established in wavelet coding and adopted in JPEG 2000, has been very successful in image coding. However, similar idea has *not* been adopted to code curves. Due to the newly proposed data structure in statistics [21] – Beamlets – it seems to be a good timing to address this problem.

Some recent papers provide good background information on the contemporary techniques. Specifically to shape coding, in a recent work, paper [69] gives a good overview on contemporary shape coding methods, which have been used in multimedia signal processing. The same authors proposed a skeleton based shape coding method. A recent paper [3] discusses an enhancement on the traditional context-based methods, which was benchmarked with CAE and JBIG [40].

In our coding method, which is named JBEAM, there are three stages. In the first stage, a beamlet representation, based on optimizing a rate-distortion function, is introduced. In the second stage, given the beamlet representation, an approach similar to the zero-tree coding is utilized to generate a symbol stream. Finally in the third stage, an entropy coding (based on Lempel-Ziv, which is available as 'gzip' in most of the UNIX systems) is applied to code a symbol stream into a bit stream.

Our coder combines three main ideas:

1. Utilizing a newly developed data structure (i.e., beamlet) in coding linear and curvilinear images.

2. A zero-tree coding strategy, which is the idea that was presented in EZW [60] and SPIHT [57] for image coders.

3. A representation, which is rate-distortion optimized.

The desired properties of a curve coder are *efficient*, *fast*, and *progressive*. "Efficient" means that for a given shape, the number of bits is as small as possible. "Fast" means that the coder requires small number of numerical operations. Being fast is an obvious requirement for any real time application. "Progressive" means that when a fraction of the coded stream is available, an approximate reconstruction of the original contour can be done. When more bits are available, the approximation becomes more faithful to the

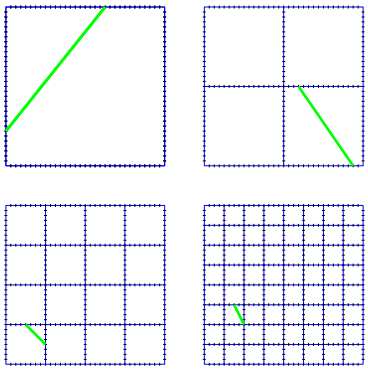


Figure 11: Dyadic squares marked with vertices, and several beamlets.

original. This property is important for image and video transmission. We will demonstrate that a beamlet based coder (i.e., JBEAM) has all the above properties.

The proposed coding method is efficient: by comparing to existing industrial standard software – JBIG – our beamlet coder requires less number of bits.

The proposed method is fast. It mainly requires a bottom-up tree pruning algorithm in computing the optimal representation. The zero-tree coding can be done efficiently. In the algorithm description, we will prove that our method has low computational complexity, essentially $O(n^2)$.

The proposed method is *partially* progressive. In each separate stage, the proposed method allows partial reconstruction. It achieves the progressivity stagewise, instead of globally.

4.2 Beamlets

The main idea of beamlet representation is to approximate linear objects in 2-D by multi-scale line segments [22]. The dictionary of beamlets has low order of complexity compared to the set of all possible line segments; and it takes a small number of beamlets to approximate an arbitrary line segment within a given precision.

In this section, we describe beamlets, its digitization, and how to code a single beamlet.

4.2.1 Beamlet Dictionary

The beamlet dictionary is a dyadically-organized library of line segments at a range of scales and locations, having different orientations and lengths. It facilitates a multiscale approximation to the collection of all line segments. Beamlets can be defined in the following three steps:

1. Recursive Dyadic Partitioning (RDP). Consider a unit square $[0, 1] \times [0; 1]$, we first divide the unit square into two by two smaller squares with equal dyadic sidelengths.

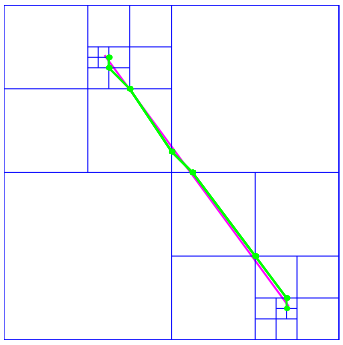


Figure 12: Approximating a line segment (red) by a chain of beamlets(green).

We then further divide the small squares into two by two even smaller squares, still having equal and dyadic sidelengths. This process is repeated until the finest scale is reached.

2. Vertex Marking. On the boundary (four sides) of each square, starting from the northwest corner, we mark vertices at equal distance. The inter-distance is fixed in advance, and it does not vary with the sidelengths of the squares.

3. Connecting. Within each square, any pair of vertices on its boundary determines a line segment. This line segment is called a beamlet.

Figure 11 illustrates the key idea in defining beamlets. The set of all beamlets forms a *beamlet dictionary*. Figure 12 shows that any line segment can be approximated by a chain of beamlets.

4.2.2 Digital Beamlets

Given two end pixels on the boundary of a dyadic square, a *digital beamlet* is determined by digitally interpolating between the two pixels. There are two topological possibilities:

1. multiple pixels are allowable per column, or

2. only a single pixel is allowable per column.

A more rigor (i.e. mathematical) discussion regarding how to numerically realize the above is provided in Appendix 4.8.1. Figure 13 gives two digital beamlets with the same endpoints, satisfying different topological conditions. We will concentrate on a digital beamlet that follows the second situation, which we will call a *thin* digital beamlet.

4.2.3 Progressive Coding of a Single Beamlet

We count possible configurations for a single beamlet. This will give us an upper bound of the number of bits that is required for coding a single beamlet. The result is summarized as follows.

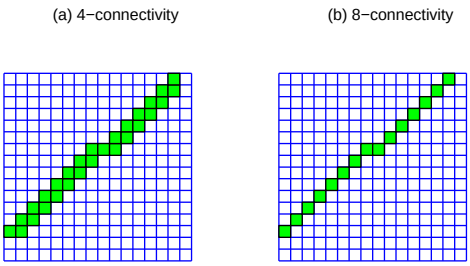


Figure 13: Two discrete beamlets. One is generated by allowing connecting to four neighboring pixels (a); the other eight neighboring pixels (b).

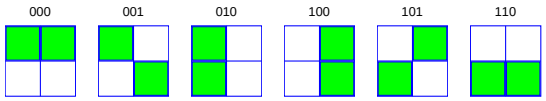


Figure 14: Six possible beamlet in a 2 by 2 image. The coded bits are in the titles.

Lemma 1. *The maximum number of bits to code a single beamlet in a* 2^s *by* 2^s *image is* $2s + 3$.

Proof. See Appendix 4.8.2. □

The upper bound given in the above lemma gives a good estimate for the number of bits that are needed. However, when the size of the square (2^s) is small, the number of bits can be significantly reduced. Table 4 summarizes these quantities. This prompts us to treat the beamlet coding for small squares differently than for large squares, as we will do in our coding scheme.

Table 4: Number of bits to code beamlet in dyadic squares with a range of sizes. It is observed that when scale s is large, the required number of bits basically follows the rule in Lemma 1. However when the scale is small, e.g. $s = 0, 1$, the required number of bits is smaller than $2s + 3$. See the numbers in the parentheses.

Scale, s	0	1	2	3	4
Size, 2^s	1	2	4	8	16
# beamlets	1	$\binom{4}{2}$	$\binom{12}{2}$	$\binom{28}{2}$	$\binom{60}{2}$
		$= 6$	$= 66$	$= 378$	$= 1770$
# bits	(1)	(3)	7	9	11

Given a 2^s by 2^s image, we would like to code all the possible beamlets progressively and within the bits limits that are given in Table 4. Let us first consider some simple cases. For a 1 by 1 image, there is only one possibility; hence one bit is sufficient. For a 2 by 2 image, there are four boundary pixels. Hence there are six possible beamlets, which are depicted in Figure 14. They can be coded by 3 bits.

It is more interesting to consider the case when $s \geq 2$. Note that when the size of an

image is 2^s by 2^s, there are $2^{s+2}-4$ boundary pixels, which can be coded by $(2s+3) = 2(s+2)-1$ bits. The following lemma appears to be a stronger result.

Lemma 2. *Consider a length-(2^K-1) consecutive sequence,$\{1,2,3,\ldots,2^K-1\}$, in which there is a substring,*

$$s(i,j) = \{i, i+1, \ldots, j-1, j\},$$

where $1 \le i < j \le 2^K - 1$. There is a coding scheme for all substrings $s(i,j)$ such that

1. *the coder is progressive;*
2. *it takes at most $(2K-1)$ bits;*
3. *when there are $2k-1 (k < K)$ bits available, the position of the two ends of interval can be recovered with worst case distortion (measured by absolute distance) 2^{K-k};*
4. *the above distortion is the minimum possible distortion.*

Proof. See Appendix 4.8.3. □

The proof of the above lemma is constructive. It leads to a progressive coding scheme for all single beamlets in a square. Recall in a 2^s by 2^s square, there are $2^{s+2}-4$ boundary pixels. Assume these boundary pixels form a sequence, for example by sampling them clockwise. Then add three dummy pixels at the end of this sequence. According to the above lemma, there is a progressive coder, whose worst case distortion is minimized for a given number of bits.

The above coding scheme is progressive: i.e., when more bits coming in, the position of a beamlet can be approximated better. To illustrate this phenomenon, we present some representable beamlets for a 8 by 8 image, when $1, 2, 3, 4$, and 8 bits are available: Figure 15-19. *Representable beamlets* are picked as follows. They are derived form the coding scheme that is described in detail in Appendix 4.8.3. Consider there is only one bit available. Suppose the upper right pixel is chosen as the 'central pixel' of the coding. The first bit will tell if this pixel is located between the two end pixels of a beamlet, or not. If not (the first bit is '0'), a representative beamlet goes from the central pixel to a last pixel. If yes (the first bit is '1'), a representative beamlet goes from the middle of the first half to the middle of the second half. These two beamlets are plotted in Figure 15. When more bits are coming in, the representation of the beamlet becomes more accurate. When there are 8 bits, the representative beamlets are illustrated in Figure 19. Note that 9 bits is sufficient to code all digital beamlets in an 8 by 8 image. Hence the beamlets in Figure 19 can approximate an arbitrary beamlets with high accuracy – more specifically, with at most 1 distortion about the positions of endpoints.

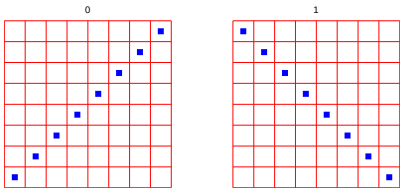


Figure 15: Representable digital beamlets in a 8 by 8 image when 1 bit is available.

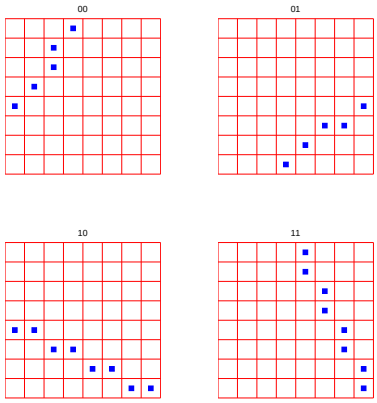


Figure 16: Representable digital beamlets in a 8 by 8 image when 2 bits are available. In this figure and Figure 17 & 18, the coded bits are in the titles.

4.3 Beamlet Representation, Distortion, and Rate

In this section, we describe beamlet representation, and how to obtain an optimal beamlet representation.

4.3.1 Beamlet Representation

We describe the basic idea of the beamlet representation. In the initial step, we build a complete beamlet-decorated quad-tree, having depth $\log_2(n)$. Each node is associated with a square, and its four child nodes are associated with its four sub-squares. A Beamlet Representation (BR) is associated with a particular image. Each BR corresponds to an admissible subtree of a complete quad-tree. Such a subtree is associated with an incomplete Recursive Dyadic Partitioning (RDP) too. For late convenience, we choose $\mathcal{P}$ to denote an incomplete RDP. Each node is marked with one of three symbols: Q – 'Quadratic Split', N – 'No Split', and B – 'Beamlet Decorated'. Symbol Q can only be associated with intermediate node, while symbols N and B can only be associated with terminal nodes. If there is no content in a subsquare in a RDP, then the node associated with this subsquare is marked by N. If a beamlet can depict the image content in a subsquare, then the associated node is marked by B. We do this marking recursively: if N or B can be assigned, the partition stops; otherwise, mark the current subsquare with Q, divide it into smaller subsquares, and repeat this process in the smaller subsquares. In fact, this coincides with the idea of

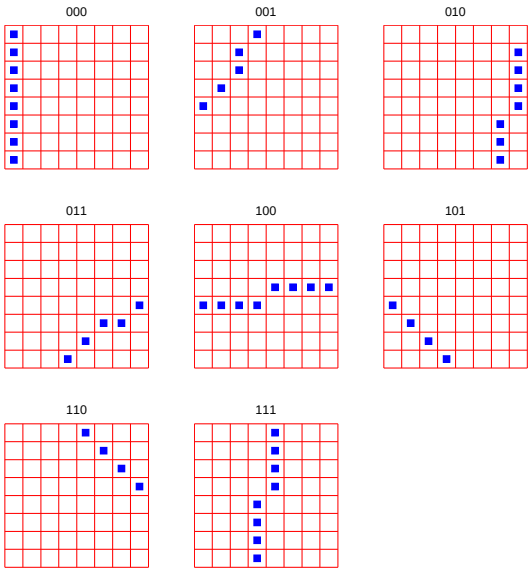


Figure 17: Representable digital beamlets in a 8 by 8 image when 3 bits are available.

Beamlet-Decorated Recursive Dyadic Partitioning (BDRDP) that is described in [21]. We utilize this structure, in order to facilitate zero-tree coding.

Given a BR, one needs to know how many bits are needed in order to code. Recall that $\mathcal{R}$ stands for a RDP; by allowing it to contain symbols, it can also denote a BR. (We allow a little bit abuse of notation.) Let $S \in \mathcal{P}$ indicates that a subsquare S is a *terminal* node in RDP $\mathcal{P}$. The overall rate function that is associated with RDP $\mathcal{P}$ can be defined as follows,

$$\begin{aligned} R(\mathcal{P}) &= \#\text{ of bits that are required to code } \mathcal{P} \\ &= [\log_2(3)](\#\text{ of symbols in a BR }) + \sum_{S \in \mathcal{P}} R(S), \end{aligned}$$

where the "# of symbols in a BR" is the numbers of 'N', 'Q', and 'B' in a beamlet representation, the constant $\log_2(3)$ is the average cost to code a symbol, and $R(S)$ is the number of bits for coding a digital beamlet if square S is marked as 'B'.

Note that it takes very little to code a symbol; while it might take quite a lot to code a beamlet. In the forthcoming zero-tree coding, we will give symbols more priority.

On the other hand, in computing the 'rate', we ignore the interdependence among the symbols, which may substantially decrease the required amount of bits. The above formula is merely an estimate. The purpose is to facilitate the computing of a Rate-distortion-optimized BR.

An example is presented in the next section to illustrate a BR.

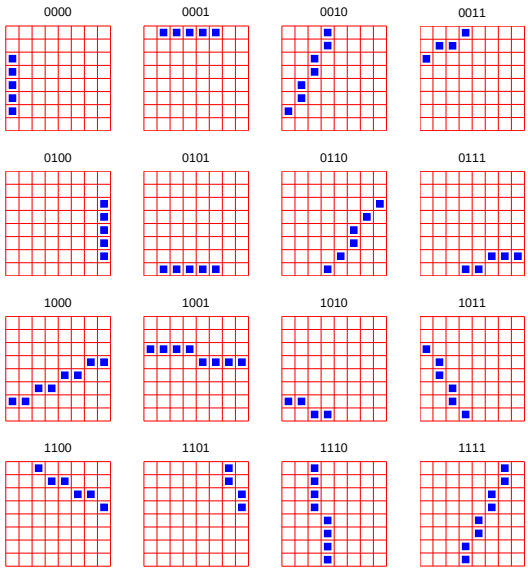


Figure 18: Representable digital beamlets in a 8 by 8 image when 4 bits are available.

4.3.2 A simple example for BR

As an example, we consider a binary image in Figure 20. At the coarsest level, no beamlet can fit well; hence a quadratic split (Q) is assigned. At the next level, two squares contains no dark pixels. They are labelled as empty (N). One square contains a beamlet (B_1). The last remaining square has no appropriate fitting; hence it is split into four (Q). In the next level, the four smallest squares either contains no dark pixels or are fitted by beamlets. The corresponding partitioning and the vertices of beamlets are given as follows.

level 0	level 1	level 2
Q		N N – –
	Q N	B_2 B_3 – –
	N B_1	– – – –
		– – – –

(23)

where Q, N, and B denote Quadratic Split, No Split, and Beamlet respectively. Symbol "–" in the above table means no need of symbols. The coordinates of beamlets and their corresponding bit representations are:

	first vertex	second vertex	binary represent.
B_1	16	45	10011111011
B_2	5	16	100100000
B_3	8	24	101101010

(24)

The bits in the last column are the results of coding single beamlets. Note that from the above two tables, (23) and (24), one can reconstruct the binary image.

In the next section, we define the distortion of a beamlet representation.

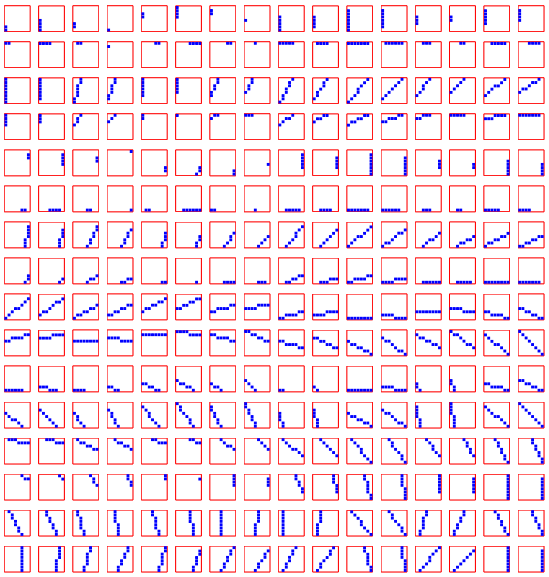


Figure 19: Representable digital beamlets in a 8 by 8 image when 6 bits are available.

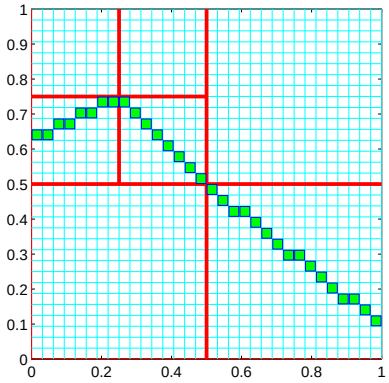


Figure 20: An example of binary image. Note this image is not a shape. It is made for illustrating the idea of beamlet representation.

4.3.3 Distortion for a single beamlet

We define a distortion function, to measure the goodness-of-fit of a beamlet in a given dyadic square.

Let y denote a set of image points: black pixels. Let S denote a dyadic square. Let $y \cap S$ denote the set of image points within S. And let b denote a beamlet, which interchangeably represents a set of pixels that the corresponding digital beamlet traverses. There are two quality measures of the beamlet approximation. One is the sum of the Euclidean distances between image points and the beamlet. The other is the number of pixels that are non-coincided – pixels that are either in the image, but not in the beamlet; or in the beamlet,

but not in the image. We define a distortion function by combining these two measures:

$$d_S(y,b) = L_1(y \cap S, b) + \lambda \cdot L_2(y \cap S, b), \tag{25}$$

where λ is a constant,

$$L_1(y \cap S, b) = \sum_{x \in y \cap S} \rho^2(x,b), \tag{26}$$

with $\rho^2(x,b)$ denote the distance between a pixel x and a digital beamlet b, and

$$L_2(y \cap S, b) = \sum_{x \in y \cap S} I\{x \notin b\} + \sum_{x \in b} I\{x \notin (y \cap S)\}, \tag{27}$$

where $I\{\cdot\}$ is an indicator function. Constant λ is prescribed and positive; it controls the tradeoff between the two types of measure.

In the above definitions, the function $L_1(y \cap S, b)$ is to measure the deviation from y to the digital beamlet b. Within S, when y and b are identical, one has $L_1(y \cap S, b) = 0$. The detailed calculation regarding $L_1(\cdot,\cdot)$ is given in Appendix 4.8.4. The second function $L_2(y \cap S, b)$ measures the degree of overlapping between the image y and the beamlet b. When the image y and beamlet b coincide in S – occupying the same set of pixels – we have $L_2(y \cap S, b) = 0$; and vice versa.

Now let $\mathcal{B}_S$ denotes the set of all possible digital beamlets in square S. The minimum possible distortion of a beamlet fit in square S is given as

$$D(y,S) = \min_{b \in \mathcal{B}_S} d_S(y,b). \tag{28}$$

4.3.4 Operational Rate-Distortion Representation

We define the overall distortion of a BR $\mathcal{P}$ as

$$D(y,\mathcal{P}) = \sum_{S \in \mathcal{P}} D(y,S),$$

where y is a binary image, and $D(y,S)$ is the square limited distortion function that is defined in (28). Recall that $S \in \mathcal{P}$ indicates that a square S is a *terminal* node in BR $\mathcal{P}$. In fact, we only need to consider the square S that is marked as Beamlet decoration (i.e. B). If square S is marked as N, then $D(y,S) = 0$, because the image is empty within S.

To find the rate-distortion optimized beamlet representation, we consider the following constrained optimization problem:

$$\min_{\mathcal{P}} R(\mathcal{P}), \qquad \text{subject to } D(y,\mathcal{P}) \le D^0,$$

where constant D^0 is the maximum allowable distortion. The above problem is normally solved via a Lagrangian Multiplier method: for $\tau > 0$, consider

$$\min_{\mathcal{P}} R(\mathcal{P}) + \tau D(y,\mathcal{P}). \tag{29}$$

A range of the values of the Lagrangian multiplier, τ, are examined. The frontier of the set, $\{(D(y,\mathcal{P}), R(\mathcal{P})) : \mathcal{P} \text{ is a RDP}\}$, which is called a *feasible set* in many publications, is explored.

In paper [61], as an image coding project, an algorithm was given to 'search for the desired R-D operating slope', which is τ. The same algorithm can be adopted here. The basic idea in their approach is to design a binary search, which is well-known in numerical computation. We skip the details in this paper.

4.3.5 Bottom-up Tree Pruning Algorithm

The problem (29) can be solved efficiently by using a *tree pruning* algorithm. Similar algorithms have been designed for other purposes, for example the Best-Orthonormal-Basis (BOB) in finding the optimal wavelet packets.

First of all, each RDP is associated with an admissible subtree of a complete quad-tree. Suppose node (square) S has four children (subsquares) $S_i, i = 1, 2, 3, 4$. For $1 \le i \le 4$, let $(D_\tau(y, S_i), R_\tau(S_i))$ denote the pair of distortion and rate, such that for a given value of τ, the objective function in (29) is minimized within square S_i. We consider the options at node S:

1. If all $S_i, i = 1, 2, 3, 4$, are marked with symbol 'N', then node S is marked with 'N', and assign
$$D_\tau(S) = 0; \quad \text{and} \quad R_\tau(S) = \log_2(3).$$

2. If node S is not marked with 'N', there are two possibilities:

   (a) If node S is marked with a 'Q', a quad-split is implemented, and we assign
   $$\begin{aligned} \tilde{D}_{\tau,1}(y,S) &= \sum_{i=1}^{4} D_\tau(y, S_i), \quad \text{and} \\ \tilde{R}_{\tau,1}(S) &= \log_2(3) + \sum_{i=1}^{4} R_\tau(S_i). \end{aligned}$$
   The objective function in (29) is equal to
   $$\widetilde{R}_{\tau,1}(S) + \tau \cdot \widetilde{D}_{\tau,1}(y,S). \tag{30}$$

   (b) If node S is marked with a 'B', a digital beamlet is fitted within square S, and we assign
   $$\begin{aligned} \widetilde{D}_{\tau,2}(y,S) &= D(y,S), \quad \text{and} \\ \widetilde{R}_{\tau,2}(S) &= \log_2(3) + (2j+3), \end{aligned}$$

where function $D(y, S)$ is the minimum distortion that is defined in (28) and j is the scale of square S: square S is a 2^j by 2^j square. The objective function in (29) becomes

$$\widetilde{R}_{\tau,2}(S) + \tau \cdot \widetilde{D}_{\tau,2}(y, S). \tag{31}$$

We pick the option that gives the minimum value between (30) and (31).

Apparently, the above provides a bottom-up tree 'pruning' algorithm. Since this is a fairly well-known algorithm, we choose to describe it briefly. Fix a value for τ. Give a quad-tree. Starting from the bottom of a quad-tree, we consider a quadruplet that share a parent. Following the above description, determine which one of the three options that shall be chosen. Record the corresponding rate and distortion, and store it at the parent node. Do this for all quadruplets at this level of the quad-tree. Then move up to the above level, repeat the procedure. The entire procedure is terminated when the top node is reached.

The resulting BR is a solution to the problem in (29); this can be proved by induction. Such a solution is often called a *dynamic programming* approach.

By running this algorithm for different values of τ, one can find a set of rate-distortion-optimized beamlet representations, having different pairs of (rate, distortion).

A recent manuscript [61] adopts a similar approach to code piecewise smooth images. Despite the strong similarity in the algorithm, their algorithm serves a different purpose: images rather than curves.

4.3.6 Computational Complexity

Suppose we have an n by n image. Suppose n is dyadic: there exists an integer J, such that $n = 2^J$. At each level of the quad-tree, there are $1, 4, 4^2, \ldots, 4^{J-1}$, number of quadruplets. Note that the sum of the above numbers is $\leq n^2$. Suppose in choosing among the three options as described in the previous section, for each quadruplet, it takes $O(1)$ operations. The overall complexity of the algorithm is at most $O(n^2)$. In fact by studying the three options in the previous section, assuming that the beamlet fitting has been pre-computed for all squares whenever it is appropriate, the complexity of choosing one option can be verified as $O(1)$.

In fact, the fitting of beamlet in a square by itself may not be an easy task. We intend to separate this issue. Because it is not the focus of this paper. Sophisticated algorithms have been designed for these problems, e.g. [8]. The main idea is to utilize or improve the Fourier Slice theorem, which is widely used in scientific fields such as computational tomography.

4.4 Coding

Given that a beamlet representation has been computed, a coding scheme is described.

4.4.1 Overview of a Beamlet Coder

There are three steps in a beamlet codec. Only the encoding part is described. Since each step of a beamlet encoder is invertible, so the decoding part is not hard to derive. A binary image that contains linear or curvilinear features is the starting point. From a binary image, a beamlet representation is calculated, as described in the previous section. This step is called *Beamlet transform.* A Beamlet Representation (BR) is generated. Based on the BR, a stream made by symbols and bits is derived. We call this stream a *symbol stream.* Finally, an arithmetic coder or an entropy coder is applied. The Table 5 describes the entire procedure.

Table 5: Overview of A Beamlet Coder

Binary Shape Image,	
↑ ↓	Beamlet Transform
Beamlet Representation,	
↑ ↓	BR to symbol stream conversion
Symbol Stream,	
↑ ↓	Coding
Bit Stream.	

4.4.2 Zero-tree Compression of Beamlet Representation

Now from two tables, (23) and (24), that are generated in the previous section, we consider how to generate a stream that is made only by symbols (Q, N, and B) and bits (0 and 1). We describe it in two stages.

In the first stage, we generate a L by $2J+4$ array, which is made by symbols and binary numbers. Here L denote the number of symbols in table (23). We first list all the symbols on the first column, in an order such that the symbols at coarser scales coming before the symbols at finer scales. The binary representation of *beamlets* are listed thereafter. For a beamlet representation given by table (23) and (24), the array looks as follows.

$$\begin{array}{lll} l_0 \rightarrow & \mathrm{Q} & \\ l_1 \rightarrow & \mathrm{Q} & \\ & \mathrm{N} & \\ & \mathrm{N} & \\ & \mathrm{B}_1 & 1\,0\,0\,1\,1\,1\,1\,1\,1\,0\,1\,1 \\ l_2 \rightarrow & \mathrm{N} & \\ & \mathrm{N} & \\ & \mathrm{B}_2 & 1\,0\,0\,1\,0\,0\,0\,0\,0 \\ & \mathrm{B}_3 & 1\,0\,1\,1\,0\,1\,0\,1\,0 \end{array} \tag{32}$$

Note a notation l_j is used to denote the starting point of symbols coming from scale j. Also

note that at the same scale, the order of symbols is not critical, but has to be fixed. We can choose any reasonable ordering scheme, e.g. raster scan [36].

In the second step, a symbol-and-bit stream is generated by the following algorithm.

Algorithm: Convert a Beamlet Representation to Symbol Stream

For $j_1 = 0 : J$, where $J = \log_2(n)$ and n is the size of the image,

1. Enumerate all symbols at level j_1, by enumerating first elements of rows between l_{j_1} and $l_{j_1+1} - 1$.

2. Enumerate bits by the following procedure.

   For $j_2 = 0 : j_1 - 1$,

   enumerate the $(j_1 - j_2 + 3)$rd and $(j_1 - j_2 + 4)$th bits of the binary representation of all beamlets in scale j_2.

   End

   Enumerate the first three bits of the binary representation of all beamlets in scale j_1.

End.

Enumerate all remaining bits, by following the order given in the second "for" loop above.

The above algorithm has appeared in EZW and SPIHT [60, 57]. Applying the above algorithm to the table in (32), the result looks like:

Q, Q, N, N, B, $B_1(1)$, $B_1(2)$, $B_1(3)$, N, N, B, B, $B_1(4)$, $B_1(5)$, $B_2(1)$, $B_2(2)$, $B_2(3)$, $B_3(1)$, $B_3(2)$, $B_3(3)$, ... $B_1(2J)$, $B_1(2J+1)$, $B_2(2J-2)$, $B_2(2J-1)$, $B_3(2J-2)$, $B_3^2(2J-1)$.

Here $B_i(k)$ denotes the kth bit of the ith beamlet. Substituting the symbols with the values from (23) and (24), the complete symbol stream is

Q, Q, N, N, B, 1, 0, 0, N, N, B, B, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, EOF.

Here "EOF" stands for the end of the file.

By using an arithmetic coder [71] or a Lempel-ziv coder, one can further code the above stream into bit stream.

It appears that the transformation (i.e. beamlet representation) and the conversion (BR to symbol stream) are decoupled. This seems to be an undesired feature of our coder. In fact, they are integrated in the sense that the conversion always leave the fine scale information at the later part of the symbol stream. The BR and the BR to symbol stream conversion are consistent.

4.5 Simulations

In this section, simulations are conducted to evaluate the performance of our beamlet coder. In the first set of experiments, Section 4.5.1, we compare JBEAM with JBIG in coding binary images, which are made by curves. The progressivity and rate-distortion behavior of JBEAM, as well as illustrative application in a video stream, are provided in the second set of experiments, Section 4.5.2.

4.5.1 Line Images

We test the performance of JBEAM on some map images, which are shown in Figure 21. They are borders of countries; each map is a 256 by 256 binary image. The original data is provided in the software package [2]. The compression ratios are compared with JBIG 2 [40].

Figure 21: Country borders. We use them as our test images.

Table 6: Simulation result for lossless coding of border maps of nine countries.

Country	Pixels	JBIG 2 (lossless)	JBEAM (lossless)	JBEAM (lossy)
Belgium	1233	7888	6114	3098
Belize	822	5768	4391	2021
Canada	2911	15224	14555	7366
Switzerland	845	6088	4171	2135
China	813	5792	4609	2041
Cuba	658	4360	3303	1542
Germany	871	5792	4916	2138
Denmark	1350	7992	6780	3735
France	920	6448	5300	2339

The experiment results are given in Table 6. In the table, the first column (Country) lists the names of the nine countries, and the second column (Pixels) lists the numbers of black pixels per image. In the third and fourth columns, the performances of JBIG 2 and JBEAM for lossless coding are compared. The last columns list the number of bits for lossy JBEAM coding. Figure 22 presents the difference between a reconstruction from a lossy JBEAM and the original binary image. It is observed that by introducing a permissible maximum error of one pixel, we are able to reduce the total number of bits by about 50%.

In fact, due to the design of JBEAM, in most of the cases, by allowing a small amount of distortion, one may significantly improve the coding efficiency. Because JBEAM is specifically designed for coding linear and curvilinear features. In each square, only *one* beamlet is allowed. One can easily suggest a malicious situation against such a design: imaging two lines, one is immediately on the top of the other. This explains, in an intuitive way, why a lossy JBEAM can be much advantageous in an arbitrary image.

The above shortcoming can be overcome by applying preprocessing. For example, one can intentionally 'thin' all the linear and curvilinear features, by calling morphological operations.

4.5.2 Coding Object Shapes in Video Sequences

To illustrate other aspects of JBEAM, we choose to apply it on a video sequence. The shapes in this sequence will be coded. Our results are based on two testing sequences.

- *Figure* sequence [1], QCIF(144×171 pixels per frame). The shape images are cropped to 128 by 128.
- *Mother-daughter* sequence, MPEG(2). Two representative shape images are cropped to 256 by 256.

A sample (which is the shape in one frame) from each of the above sequences is shown in Figure 23.

4.5.2.1 Progressivity of JBEAM

The progressivity of JBEAM is illustrated in Figure 24, for the first frame of the video sequence "Figure". In the beamlet representation, the parameters are chosen as $\lambda = 0.3$ and $\tau = 10$. The coded bit stream is truncated in the tails, and an approximate shape is reconstructed, based on the partial stream. In Figure 24, from left to the right, when more bits are available, the reconstruction becomes more alike to the original shape.

4.5.2.2 Rate-distortion Tradeoff

In the beamlet representation, the depth of partition mainly depends on the Lagrangian multiplier τ. In Figure 25, we fix different values of τ's, and plot the beamlet representations

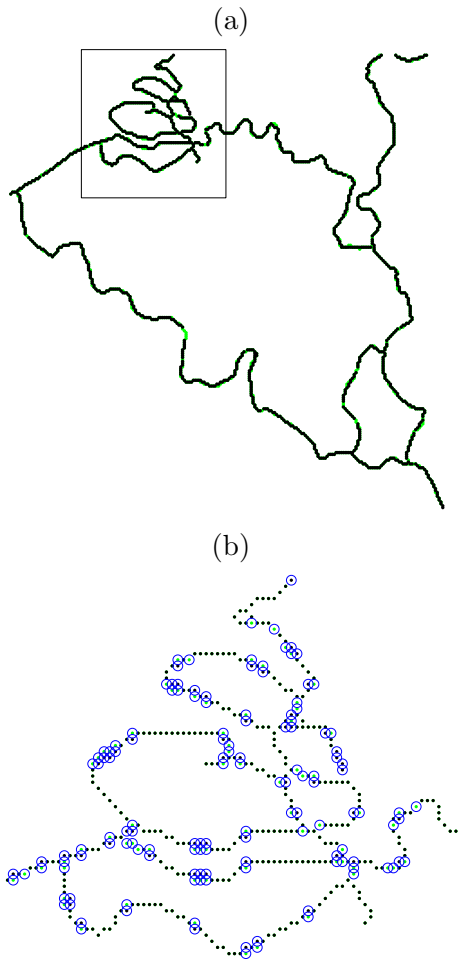


Figure 22: (a) Difference between original image and the reconstruction from a lossy JBEAM. (b) Enlargement of the most different part, which also is the hardest part to code.

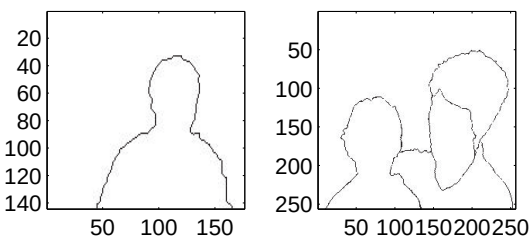


Figure 23: Shape images. Left: *Figure* sequence, Right: *Mother-daughter* sequences.

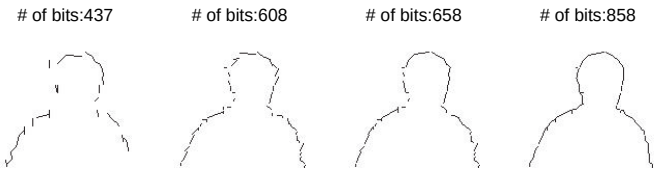


Figure 24: Progressive reconstruction of the sequence *Figure* (frame 0). The number of transmitted bits are shown in the titles.

(lower row) with their corresponding reconstructed images (upper row).

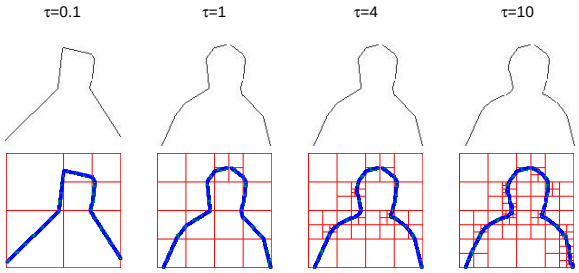


Figure 25: Beamlet representations of sequence *Figure* (frame 0), with different given τ's.

4.5.2.3 Rate-Distortion Curve

In figure 26, we plot the rate and distortion distribution for the first 50 frames of the two sequences. Figure 27 shows the rate-distortion curves of the beamlet coding averaged over these 50 frames.

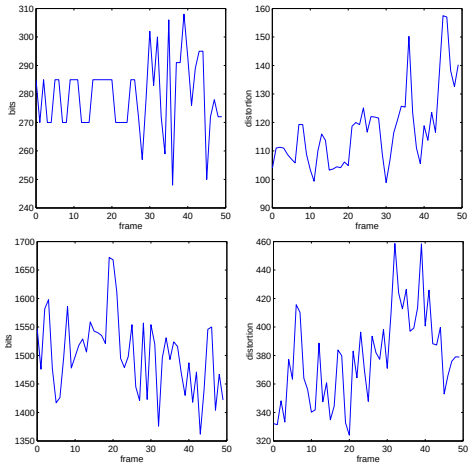


Figure 26: Rate and distortion distribution. Top: "Figure" sequences; Down: "mother" sequences.

It empirically shows that the rate-distortion curve behaves exponentially.

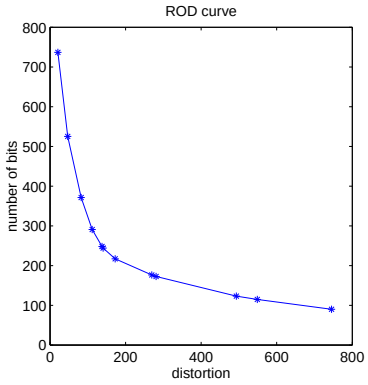


Figure 27: Rate-distortion curves of two sequences.

4.6 Discussion

4.6.1 General Properties

The proposed approach is fast. In computing the optimal beamlet representation, a bottom-up tree pruning algorithm can be used. It is proven that it can be an $O(n^2)$ algorithm, given that the beamlet fitting has been pre-computed. The beamlet fitting itself is a delicate problem. We tend to leave it for elsewhere. The zero-tree coding can be done efficiently.

The proposed method is easy to implement. The bottom-up tree pruning can be easily programmed. The zero-tree coding is easy to be programmed as well. Readers can read our published source codes (in MATLAB) to see what has been done.

The proposed method is easy to analyze. We will present some conjectures in the next section. They are easy to be proven. However, since it is not the purpose of this paper, we leave the details for somewhere else.

4.6.2 Asymptotic Analysis

The asymptotic behavior of JBEAM is of interests. Similar results have been done by motivating works, e.g. [20] and [61]. However, the targets of analysis is very different: piecewise smooth images rather than curves. In addition, the distortion function is different. We present the following results, without proofs.

1. If an image is from a polygonal model, or has finite number of parameters, or is made by a fixed number of (digital) line segments, then an oracle can achieve an exponential rate-distortion performance, i.e.

$$D(R) \sim C_1 2^{-C_2 R}, \tag{33}$$

where C_1 and C_2 are two constants, and the rate and distortion function are denoted by R and $D(\cdot)$ respectively.

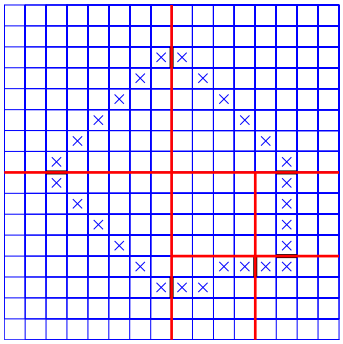


Figure 28: Illustrate a new indexing scheme of beamlets, in a beamlet representation, which facilitates coding continuous curves.

2. In the above model, a beamlet-based coder is 'near' optimal:

$$D(R) \sim C_3 2^{-C_4\sqrt{R}}, \tag{34}$$

   where C_3 and C_4 are two constants. Note that there is a square root in the exponent.

3. In a Horizon model (with $\mathbf{C}^2$), both a chaining coder and a beamlet coder can achieve the rate-distortion performance:

$$D(R) \sim \log_2^2(R) \cdot R^{-2}. \tag{35}$$

   If some one decides to use chain coding, then a beamlet coder gives the same rate-distortion performance. Note that a chain coding does not necessarily give the best rate-distortion behavior. In the wavelet literature, the above rate can be improved to $D(R) \sim R^{-2}$.

4.6.3 Variation 1: Shape Coding

JBEAM serves for a broader purpose: coding curves, which include not only shapes, but also others. In Figure 24, one can observer that the partial reconstructions in JBEAM are not satisfactory: they are *not* continuous, while shapes should be continuous. With a slight modification in the beamlet representation, one can generate a coder that is specifically designed for shapes. The key idea is to use the sides of boundary pixels, not the boundary pixels, to represent the positions of beamlets. But doing so, one enforces that the output of a beamlet representation is a continuous chain of beamlets, i.e. a continuous curve. Due to space, we have to omit the details of such a coder. Figure 28 gives an illustration of the above idea. The figure illustrates that the thickened sides of boundary pixels serve as beamlet indices in BR. The remaining components in a coder follow a nearly identical approach.

4.6.4 Variation 2: Totally Progressive?

The rate-distortion-optimization is designed for beamlet representation. It has not been integrated with coding. So far representation is separated from coding. In the problem of wavelet coding, the problem is more straightforward; hence the rate-distortion consideration is integrated with the zero-tree coding. It is not clear how such an integration can be achieved in a beamlet based coding. The challenge is that in a beamlet coding, when distortion changes, the geometry (which is represented by the structure of the quad-tree, as recorded in (23) and (24)) can be changed as well; while in wavelet coding, the quad-tree structure is fixed. Making the coder a more coherent one is an interesting future research.

4.6.5 Comparison between Lossy JBEAM and Lossy JBIG

Notice that we did not compare a lossy JBEAM with a lossy JBIG2. JBIG2 gives a wide variety of possible approaches [40]. Methods that are handily comparable were described in [45]. However, there are two difficulties. First of all, the definition of distortion is not unified. In many variations in JBIG 2, the distortions were defined to be consistent with the specific computational algorithms. The second difficulty is the lack of publicly available non-commercial software. The main purpose of this chapter is to demonstrate the feasibility of having a progressive coder based on beamlets. In this sense, a direct comparison is not essential.

4.6.6 Distortion

The distortion that was defined in Section 4.3 combines two considerations: the total distance to a beamlet, and the degree of overlapping. Discussion on the distortion for shape coding can be found in [45]. Our definition is slightly different. However, they are close to each other. We think our distortion is more intuitive.

4.6.7 Hidden Markov Model

Some beamlets may be more likely to be included in a BR than other beamlets. By integrating a statistical model, such as a Hidden Markov Model that has been used in modelling wavelet coefficients, in the computing of BR, one may derive a more efficient curve coder. Notice that in JBEAM, even though zero-tree coding is applied, which hopefully explore such an interdependence to some extent we still believe that a more direct approach is likely to outperform JBEAM. This is a promising research direction.

4.7 Conclusion

Utilizing a multiscale structure for line segments – beamlet – a zero-tree like coding method is designed for generic curves. The designed method is progressive and easy to implement.

Numerical experiments demonstrate an advantage over existing curve coding software – JBIG. A more important purpose is to describe in detail how to coding single line segments (single beamlets) and how to unify them through zero-tree coding, so that any curve can be coded. The overall approach is novel and has not been studied well in the literature. We release our software, in the hope that more research activities can be generated.

4.8 Appendices

4.8.1 Mathematical formula for a digital beamlet

Without loss of generality, we assume that (x_1, y_1) and (x_2, y_2) are the indices of two end pixels, satisfying

- $0 < x_1 < x_2 \leq n, 0 < y_1 < y_2 \leq n$, where the size of the image is n by n, and
- $|x_2 - x_1| \geq |y_2 - y_1|$.

Note that the absolute values in the last inequality is not required.

In the first situation, in which multiple pixels per column are allowed, one basically enumerates all pixels whose interiors intersect the line that is from $(x_1 - 0.5, y_1 - 0.5)$ to $(x_2 - 0.5, y_2 - 0.5)$. In the second situation, in which there is at most a single pixel per column, the set of pixels that are included in a digital beamlet can be derived as

$$\{(j, y(j)) : x_1 \leq j \leq x_2\},$$

where

$$y(j) = \left\lceil y_1 + \frac{j - x_1}{x_2 - x_1}(y_2 - y_1) \right\rceil.$$

It actually generates the same set of pixels as a well-adapted line drawing method – Bresemham's [63] line drawing algorithm.

4.8.2 Proof of Lemma 1

Proof. A digital beamlet is determined by its two end pixels. Consider a 2^s by 2^s image. The number of boundary pixels is $4 \cdot (2^s - 1) = 2^{s+2} - 4$. Hence the number of beamlets is

$$\#(\text{beamlets}) = \binom{2^{s+2} - 4}{2} = (2^{s+1} - 2)(2^{s+2} - 5).$$

The number of bits that are necessary to code the above configurations is

$$\begin{aligned} \#(\text{bits}) &= \log_2(\#\text{beamlets}) \\ &= 1 + \log_2(2^s - 1) + \log_2(2^{s+2} - 5) \\ &\leq 1 + s + (s + 2) = 2s + 3. \end{aligned}$$

□

4.8.3 Proof of Lemma 2

Proof. When $K = 2$, we have a length-3 sequence. Let the first bit indicate whether the center element, '2', is inside the substring or not. If it is, then the two more bits will code whether the remaining two elements are inside the substring or not; if the center element is *not* inside the substring, then it takes one more bit to code whether the *prior* or the *post* element forms a single-element substring. In summary, the Lemma holds at $K = 2$.

Now induction is used. Assume that when $K = k_0$, there is a coding scheme that satisfies the conditions in Lemma 2. Now consider $K = k_0 + 1$. Element 2^{k_0} becomes the central element. Use the first bit to code whether the central element is inside the substring or not. There are two cases:

1. If the central element is inside the substring, there are 2^{k_0} possibilities for the beginning and the end of the substring – including the central element itself – hence it takes $2(k_0)$ bits to code. Overall, it takes $2(k_0) + 1$ bits.

2. If the central element is *not* inside the substring, it takes one bit to code which side of the central element (i.e., before or after) the substring occurs. After this, it becomes a coding substring problem for a length-$(2^{k_0} - 1)$ sequence, for which there is a progressive coding scheme that takes no more than $2k_0 - 1$ bits. Overall, the coding scheme takes no more than $2k_0 - 1 + 2 = 2(k_0 + 1) - 1$ bits.

This proves the first two statements.

To prove statement 3), due to the inductiveness of the above construction, we only need to prove that when 2 bits are available, the absolute distance of the end positions can be limited at 2^{K-1}. This is evident because there are $2 \cdot 2^{K-1} - 1$ elements. By knowing if the middle element is inside the interval (based on the first bit) and the side of the interval (second bit), there are 2^{K-1} possible positions of the boundary elements. Hence we proved 3).

To prove 4), we use a standard argument from Information Theory, namely Kolmogorov entropy. Consider all the possible pairs (i, j), which is the starting and end position of an interval. We have $1 \leq i < j \leq 2^K - 1$. These pairs form the upper triangle of a $(2^K - 1)$ by $(2^K - 1)$ $2 - D$ array, with (i, j) being the coordinates of the points. A coding scheme with maximum-absolute-distance distortion can be viewed as covering such a triangular region with squares, having the same side-length. To achieve 2^{K-k} distortion, the side length of these squares must be 2^{K-k}. Simple calculation will reveal that it takes at least $(2^k)^2/2 + 2^k/2 = 2^{2k-1} + 2^{k-1}$ squares to cover the entire triangle. The required number of bits is given by $\lceil \log_2(2^{2k-1} + 2^{k-1}) \rceil = 2k$.

From all the above, the proposition is proved. □

4.8.4 A Tune-up

The point-to-beamlet distance $\rho_1(x, b)$ in (26) can be computed as follows. Note that further discussion on refining this function will be in Section 4.8.4. Let (x_1, y_1) and (x_2, y_2) be the two end pixels of beamlet b. Imagine a continuous beamlet, which is a straight line between two points: $(x_1 - 0.5, y_1 - 0.5)$ and $(x_2 - 0.5, y_2 - 0.5)$. Consider another pixel x with index (x_3, y_3). The distance between the pixel x and beamlet b is the Euclidean distance from the center of the pixel x – $(x_3 - 0.5, y_3 - 0.5)$ – to the straight line given by points $(x_1 - 0.5, y_1 - 0.5)$ and $(x_2 - 0.5, y_2 - 0.5)$. The distance $\rho_1(p, b)$ can be computed by

$$\rho_1(x, b) = \frac{2 \cdot |\det(\Delta)|}{|b|}, \tag{4.8.36}$$

where

$$\begin{aligned} \det(\Delta) &= \frac{1}{2} \begin{vmatrix} 1 & 1 & 1 \\ x_1 - 0.5 & x_2 - 0.5 & x_3 - 0.5 \\ y_1 - 0.5 & y_2 - 0.5 & y_3 - 0.5 \end{vmatrix} \\ &= \frac{1}{2} \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{vmatrix}, \end{aligned}$$

which is the signed area of a triangle determined by points $(x_i, y_i), i = 1, 2, 3$; and $|b|$ is the length of a line segment between two points $(x_i, y_i), i = 1, 2$:

$$|b| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

The above procedure can be justified as follows. Notice that in the right hand side (R.H.S.) of (4.8.36), the numerator is twice of the area of the triangle that is determined by the line segment b and the point x; in addition, the denominator is the length of the beamlet b. Hence while the beamlet b is considered as a base, the ratio (in the R.H.S.) is the height of the triangle, which is equal to the point-to-line distance between the point x and the beamlet b.

When pixel x is on a digital beamlet b, one should expect $\rho_1(x, b) = 0$. In order to achieve this, a decision rule is designed before applying the formula in (4.8.36). The intuition is that if the continuous beamlet –which is a line segment–traverses a square associated with a pixel, then the distance $\rho_1(x, b)$ is set to zero.

Following the previous notations, let (x_3, y_3) be the index of pixel x. Let (x_1, y_1) and (x_2, y_2) be the two endpoints of a digital beamlet b. The continuous beamlet is a line segment that connects two points: $(x_1 - 0.5, y_1 - 0.5)$ and $(x_2 - 0.5, y_2 - 0.5)$. The pixel x is associated with a square that is centered at point $(x_3 - 0.5, y_3 - 0.5)$ with four corners:

$(x_3 - 1, y_3 - 1)$, $(x_3, y_3 - 1)$, (x_3, y_3), and $(x_3 - 1, y_3)$. This square does *not* intersect the continuous beamlet iff the following four quantities have the same sign:

$$
\begin{aligned}
\text{(V1)} &= \begin{vmatrix} 1 & 1 & 1 \\ x_1 - 0.5 & x_2 - 0.5 & x_3 - 1 \\ y_1 - 0.5 & y_2 - 0.5 & y_3 - 1 \end{vmatrix} \\
&= \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 - 0.5 \\ y_1 & y_2 & y_3 - 0.5 \end{vmatrix}, \\
\text{(V2)} &= \begin{vmatrix} 1 & 1 & 1 \\ x_1 - 0.5 & x_2 - 0.5 & x_3 \\ y_1 - 0.5 & y_2 - 0.5 & y_3 - 1 \end{vmatrix} \\
&= \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 + 0.5 \\ y_1 & y_2 & y_3 - 0.5 \end{vmatrix}, \\
\text{(V3)} &= \begin{vmatrix} 1 & 1 & 1 \\ x_1 - 0.5 & x_2 - 0.5 & x_3 \\ y_1 - 0.5 & y_2 - 0.5 & y_3 \end{vmatrix} \\
&= \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 + 0.5 \\ y_1 & y_2 & y_3 + 0.5 \end{vmatrix}, \quad \text{and} \\
\text{(V4)} &= \begin{vmatrix} 1 & 1 & 1 \\ x_1 - 0.5 & x_2 - 0.5 & x_3 - 1 \\ y_1 - 0.5 & y_2 - 0.5 & y_3 \end{vmatrix} \\
&= \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 - 0.5 \\ y_1 & y_2 & y_3 + 0.5 \end{vmatrix}.
\end{aligned}
$$

Simplify the above functions will render efficient algorithm to determine whether x is on the beamlet b. The function $\rho_1(x, b)$ is modified as follows:

$$
\rho_1(x, b) = \begin{cases} 0, & \text{if quantities (V1), (V2), (V3),} \\ & \text{and (V4) have the same sign,} \\ (4.8.36), & \text{otherwise.} \end{cases}
$$

4.8.5 Sub-additivity

In beamlet approximation to a binary image, when one moves from a fine scale to a coarse scale, the minimum possible distortion should be non-decreasing, because during this procedure, more restrictions are imposed on the beamlet representation.

Let $\mathcal{B}_S$ denote the set of all beamlets in the square S. The minimum possible distortion of a beamlet fit within S is defined in (28). Suppose square S can be equally partitioned into 2 by 2 subsquares:

$$S = S_1 \cup S_2 \cup S_3 \cup S_4. \tag{4.8.37}$$

A 'reasonable' distortion function should render:

$$D(y, S) \geq \sum_{i=1}^{4} D(y, S_i). \tag{4.8.38}$$

Using the distortion function in Section 4.3, one can prove the following result.

Lemma 3. *Given the distortion function that is defined in (25), for the minimum distortion (which is defined in (28)) and an equal partition of a square S (which is described in (4.8.37)), following inequality is true:*

$$D(y, S) + \varepsilon(S) \geq \sum_{i=1}^{4} D(y, S_i), \tag{4.8.39}$$

where $\varepsilon(S)$ is a constant that only depends on the size of square S. In most of the cases, the inequality in (4.8.38) will be observed.

Proof. Let $b_0 \in \mathcal{B}_S$ and $b_i \in \mathcal{B}_{S_i}$ be the digital beamlets such that for a binary image y,

$$\begin{aligned} D(y, S) &= d(y, b_0), \qquad \text{and} \\ D(y, S_i) &= d(y \cap S_i, b_i), \quad i = 1, 2, 3, 4. \end{aligned}$$

We can easily see the following:

$$\begin{aligned} & D(y, S) \\ = \; & d(y, b_0) \\ = \; & L_1(y, b_0) + \lambda \cdot L_2(y, b_0) \\ = \; & \sum_{i=1}^{4} \tilde{L}_1(y \cap S_i, b_0) \\ & + \lambda \sum_{i-1}^{4} L_2(y \cap S_i, b_0 \cap S_i). \end{aligned} \tag{4.8.40}$$

The first equation is due to the definition of b_0. The second equation is due to the definition of the distortion function. The last one can be verified by considering the definition of the functions $L_1(\cdot, \cdot)$ and $L_2(\cdot, \cdot)$.

The difference between function $\tilde{L}_1(y \cap S_i, b_0)$ and function $L_1(y \cap S_i, b_0 \cap S_i)$ is that the former uses a line segment that is determined by a coarser scale digital beamlet b_0, while the latter uses a line segment that is determined by a set of pixels within the same square – $b_0 \cap S_i$. The latter is what has been defined for the function $L_1(\cdot, \cdot)$. Let

$$\varepsilon(S) = \max_{y \subset S} \left\{ \sum_{i=1}^{4} L_1(y \cap S_i, b_0 \cap S_i) - \sum_{i=1}^{4} \tilde{L}_1(y \cap S_i, b_0) \right\}.$$

The above equation is well defined, because the beamlet b_0 is determined by image y. Hence

$$\begin{aligned} (4.8.40) \quad &\geq \sum_{i=1}^{4} L_1(y \cap S_i, b_0 \cap S_i) - \varepsilon(S) \\ &\quad + \lambda \sum_{i-1}^{4} L_2(y \cap S_i, b_0 \cap S_i) \\ &\geq \sum_{i=1}^{4} d(y \cap S_i, b_i) - \varepsilon(S) \\ &= \sum_{i=1}^{4} D(y, S_i) - \varepsilon(S). \end{aligned}$$

Hence we proved the inequality in (4.8.39).

The function $\varepsilon(S)$ depicts the upper bound of an increment of the function $L_1(\cdot, \cdot)$ when refining a digital beamlet. Notice that such an increment is normally small. Moreover, the reduction of the distortion by introducing digital beamlets at finer scale is more likely larger than this increment. Hence in general, we may observe (4.8.38). □

CHAPTER V

DETECTION

In this chapter, we will propose a general framework – Significance Run Algorithm (SRA) for filament detection in imagery data. In section 5.1, the background and literature are summarized. In section 5.5, the distribution of the length of the longest significance run is studied. The applications to the detection problems in noisy images and textured background are proposed in section 5.3 and section 5.4, respectively.

5.1 Introduction

Detection is a fundamental problem in many applications. There are two levels of detection problems. At the first level, one may simply determine whether or not an underlying object exists. At the second level, given that an underlying object is present, one wants to estimate the location, sizes, strength, orientation, etc., of the underlying object. The second level detection problem usually is also called an *inference* problem. In this chapter, we consider the detection problem at the first level. The improvement is that by organizing the potential underlying objects, a 'most powerful' hypothesis testing can be (in some sense approximately) realized.

Recently, there has been significant advances in mathematical statistics on this problem. Namely, in [6], it is proven (as one of many cases) that when the underlying features is a digitized line segment, when the amplitude of the signal residing on the underlying feature is strong enough, there is a statistical means to reliably detect it. If the amplitude of the signal is not strong enough, which can be quantified, then there is no method that can detect it. Moreover, there is an algorithm that achieves the lowest possible order of complexity. The fundamental tools are introduced in [22].

To detect a filamentary feature, which is more likely to be *curvilinear*, the previous result, which requires *straight* line segment, is not strong enough. Very interestingly, in [5], an approach called Multiscale Significance Run Algorithm (MSRA) is developed to detect concentration of data on a curvilinear structure in uniformly distributed random point clouds. The algorithm is based on the following phenomena:

1. When there is *no* underlying object, some statistics associated with the basic objects can 'hardly' exceed a prescribed threshold. We say a basic object becomes significant if and only if its statistic exceeds the threshold. Consequently it will be 'rare' to see a chain of continuous significant basic objects.

2. When there is an underlying object and the associated amplitude A_n is 'significantly' large, the statistics on the basic objects that 'mostly' overlaps with the underlying feature are most likely to exceed a prescribed threshold. Consequently it is likely to have a 'long' chain of continuous and significant basic objects.

The detection statistic is based on the length of the *longest significance run*. The key result is that by applying MSRA, the fundamental threshold of the detectability—above this threshold, an object is detectable; below this threshold, no one should be able to distinguish it from pure noises—is obtained.

In this chapter, we will study the problem of filament detection in imagery data. This problem has a lot of applications in different areas. We give some examples as follows.

Example 1 (Ship Wake Detection). *An example of ship detection is presented in Figure 29. In the first figure, there are two small boats (with ship wakes) in the northwest corner. The second image only contains the ocean—no ship. We are interested in whether the two images are distinguishable. If they are, in what sense?*

Both images are taken by satellite, and are downloadable on the web:

- *The image (a) is taken near golden gate bridge, San Francisco, CA. Besides the two small boats, a large one left the picture, and an incoming one is near the bottom. It was taken on July 10, 1993, and covers roughly a region of* 800 *meters by* 600 *meters.*
- *Image (b) is an ocean image with no boat in it. It is taken about* 17 *km South of St. Petersburg, Florida, United States.*

Example 2 (Cryo-EM Images). *Figure 30 presents an image in Microtubule Studies that is generated by cryo-electron microscopy (cryo-EM) (http://cryoem.berkeley.edu/). Since the image is taken at molecular level, significant amount of noises are present. The question is whether at the current resolution, enough information is available for confident inference of the underlying structures.*

Example 3 (Features in Texture Images). *Figure 31 illustrates some features embedded in texture images. Again, we would like to detect the presence and locations of the embedded objects.*

We will first extend the MSRA algorithm to solve detection problems in imagery data. When the image data are a realization of Gaussian Random Field, the method in [5] can be directly adapted to solve the problem. In more general, we study the difficult problem of detecting features hidden in a homogeneous textured background. The ideas from PCA and manifold theory are used to define a detection and estimation statistics. These applications of MSRA involves the parametrial selection to optimize the power of the test. This motivates the study of the length of the longest significance run on a Bernoulli net. We will also provide a simulation approach for choosing parameters.

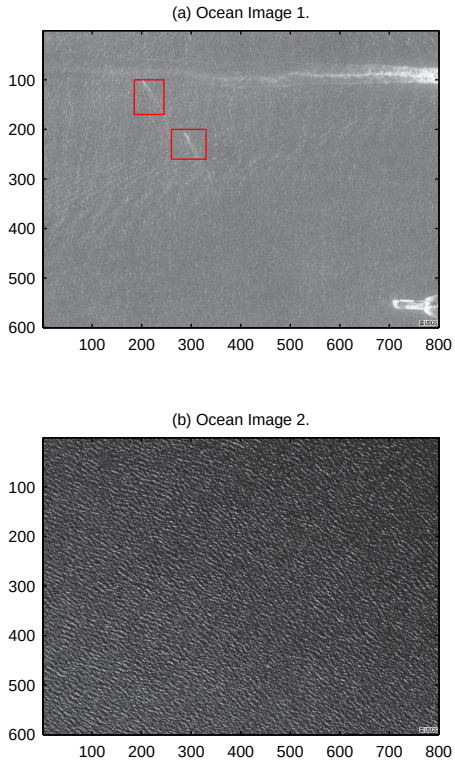


Figure 29: A ship wake image and an oceanic image. In (a), two small boats are in two boxes.

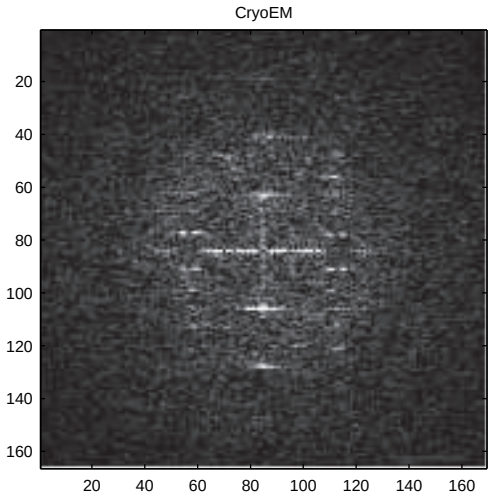


Figure 30: A cryo-electron microscopy (cryo-EM) image.

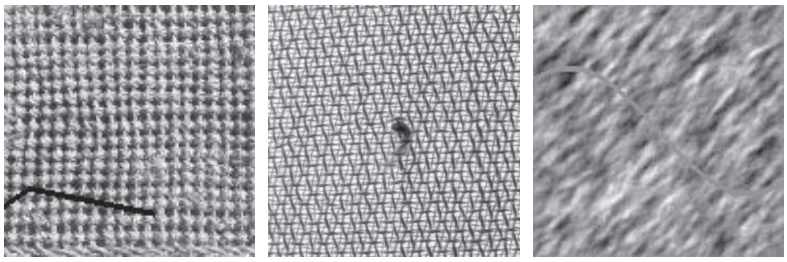


Figure 31: Features embedded in texture images.

5.2 The Significance Run Algorithm

We summarize the basic idea of the MSRA algorithm as follows.

1. *Basic elements.* Some regions, which satisfy specific geometric conditions, are chosen as *basic elements.* In [5], the basic elements are *axoids.* In multiscale approach, basic elements are the set of multiscale features (such as dyadic intervals, or dyadic rectangles, or beamlets, etc as in [6]).

2. *Significant elements.* For a given basic element, one would like to decide whether or not the content inside this element is likely to be 'abnormal': likely to overlap with an underlying structure. An abnormal basic element is *significant.* Let "p" denote the probability that a basic element is significant.

3. *Significant graph and a Bernoulli table.* If two basic elements are next to each other, they are 'connected'. Let all the significant elements be the nodes in a graph. There is an edge between two nodes iff the two corresponding basic elements are connected. Hence we have a graph, which is called *significant graph.* In the cases that are described in [5], the basic elements are equally spaced at x and y coordinates and orientations, the significance graph can be abstractized as a table; each entry has a fixed chance to be significant. For obvious reasons, this table is called a Bernoulli net (or table). An illustration of a Bernoulli net is given in Figure 32.

4. *Longest runs.* A *significance run* is a chain of connected significant nodes. The *longest run* is the longest significance run, i.e. the longest chain of connected significant basic elements in a Bernoulli net.

5. *Test.* The test statistic is the length of the longest run(s). The motivation is that if there is no underlying structure, the length of the longest run is statistically upper bounded. If the *observed* length of the longest run is much bigger than the just mentioned upper bound, then there is an evidence that there is an underlying object.

Figure 33 presents a showcase of SRA, for imagery data.

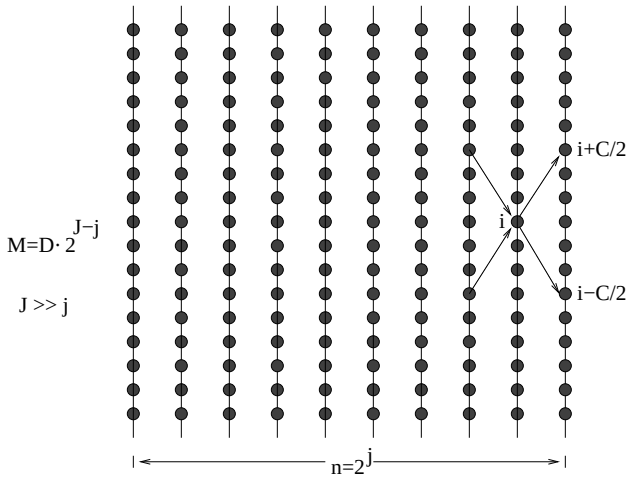


Figure 32: A Bernoulli net (significance graph).

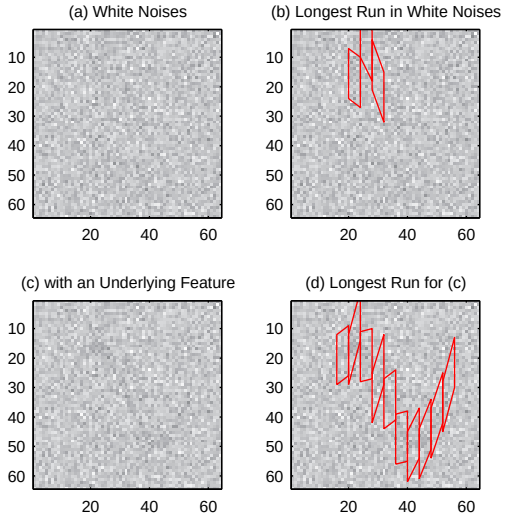


Figure 33: Two showcases of SRA. In each case, (a) is a pure noisy image; (b) is a longest run in the noisy image (a); (c) is a noisy image with an embedded feature; (d) is a longest run for image (c). The longest runs are dramatically different when there is a feature embedded.

5.3 Detection in Noisy Images

we now apply the MSRA algorithm to detect the presence of a filamentary feature in digital noisy images. This method is directly inspired by a very similar approach in [5].

5.3.1 Formulation

Image data are $x(i,j), 1 \leq i,j \leq n$, where one normally assume that n is dyadic: $n = 2^J$. The value of $x(i,j)$ is equal to the intensity at pixel (i,j).

An underlying feature can be considered as a subset of pixels: $F = \{(i,j)\}$. Let $|F|$ denote the cardinality of the set: i.e. the number of pixels in F. This subset of pixels will follow some conditions. For example, the following is an analogy of the Lipschitz condition for continuous functions.

Condition 4. *There exist C and M, such that for pixels (i_1,j_1) and (i_2,j_2), $|i_1 - i_2| < M$, we have*

$$\max_{j_1,j_2}\{|j_1 - j_2| : (i_1,j_1),(i_2,j_2) \in F\} \leq C|i_1 - i_2|.$$

Apparently, the above condition only allows the height of a feature to be 1. The above condition can be generalized to include the feature whose height (which sometimes is called thickness) is larger than 1, by considering the lower and upper boundary of the feature. In this section, we leave it as an exercise for readers.

The image data can be considered as a realization of a Gaussian random field, namely:

$$x(i,j) \sim \begin{cases} N(0,1), & \text{when } (i,j) \notin F, \\ N(A_n,1), & \text{when } (i,j) \in F, \end{cases} \tag{5.3.41}$$

where A_n is a constant.

The fundamental questions are what the pair (F, A_n) should be so that the underlying feature becomes detectable, and if it is detectable, what an efficient numerical method will be?

We would like to make a few comments before describing the data structure.

1. First of all, for point processes, the above problems (from a theoretical point of view) has been answered in [7] and [5].

2. More general conditions can be introduced, to cover more generic digital curves. For example, Condition 4 mimics the Lipschitz condition. We may later introduce a condition which mimics the Hölder condition.

3. This section will emphasize the computational framework, instead of the theoretical aspect of this problem.

4. In many existing works, the feature F is introduced as a discretization of a continuous object. For simplicity, we avoid that kind of description.

5.3.2 Digital Axoids, Significance Graph, and the Longest Paths

5.3.2.1 Motivation

Recall that a fundamental idea that is embedded in the approach of [7] and [5] is to design some basic objects, such that an underlying feature can be efficiently covered by a small set of these basic objects. In [7], these basic objects are rectangles or slanted rectangles. In [5], they are dyadic axoids (which is also called strips).

If one considers a straight line feature, then a beamlet-like scheme that is described in [22] and [6] can provide an efficient solution. However, this approach fails when the underlying feature could be curvilinear. To remedify this, the approach in [5] is to consider basic objects at a range of scales. It is proven that there exists a scale at which there is a 'good covering' of a potentially curvilinear feature. And then a longest path statistic can be used to detect the presence. Note that at different scales, the basic objects have different width and height (i.e., thickness). This is indeed a multiscale methodology. It is shown in [5] that such an approach can cope with a range of smoothness (or regularity) conditions for the underlying features. Comparing with a beamlet-like method, the new method can be considered as a method that uses thickened beamlets.

In the next section, we describe our basic objects: *digital axoids*.

5.3.2.2 Digital Axoids

As mentioned earlier, the digital axoids are derived from the dyadic strips in [5].

First of all, a scale $s, s = 1, 2, \ldots, J$, is fixed. Practically, we may impose $1 < s$ and $s < J$.

The $n \times n$ image is divided into 2^s vertical strips; each strip is an $n \times 2^{J-s}$ subimage. Let $k = 1, 2, \ldots, 2^s$, be the index of these vertical strips. For a fixed k, let $\ell_0 (\ell_0 = 1, 2, \ldots, n)$ denote a row number in the vertical strip, Let $\ell_0 + \ell_1$ ($1 \le \ell_0 + \ell_1 \le n$, and $-S \cdot 2^{J-s} \le \ell_1 \le S \cdot 2^{J-s}$, where S is a prescribed constant) denote another row number. Between pixels $((k-1) \cdot 2^{J-s} + 1, \ell_0)$ and $(k \cdot 2^{J-s}, \ell_0 + \ell_1)$, there is a digital line determined by the Bresenham's line drawing algorithm. Let $B(\cdot)$ denote a function such that the pixel set

$$\{(i, B(i)) : (k-1) \cdot 2^{J-s} + 1 \le i \le k \cdot 2^{J-s}\}$$

are the pixels that made the Bresenham's line. A *digital axoid* is a set of pixels that thickens such a line, formally, the following set of pixels

$$a(s, k, \ell_0, \ell_1) = \left\{ (i, j) : \left| \begin{array}{l} (k-1) \cdot 2^{J-s} + 1 \le i \le k \cdot 2^{J-s}, \\ \max\{1, B(i) - 2^{s-1}\} \le j \le \min\{B(i) - 1 + 2^{s-1}, n\} \end{array} \right. \right\}$$

form a *digital axoid*.

In Figure 34, digital axoids (for an 8×8 image) at different scales and positions are presented.

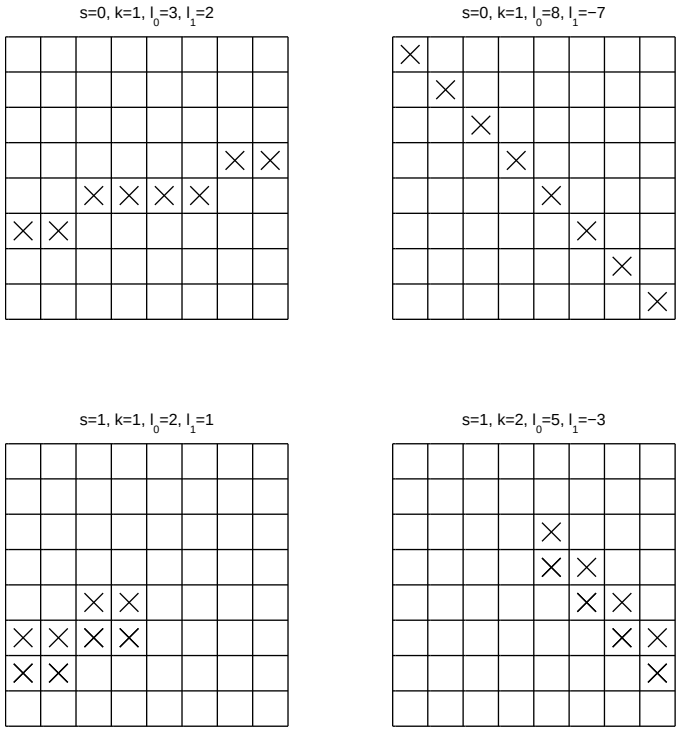


Figure 34: Four digital axoids. The marked (by $\times$) squares are pixels belonging to the digital axoids. The indices of these digital axoids are in the titles.

5.3.2.3 Significance Graph

For a given digital axoid $a(s, k, \ell_0, \ell_1)$, a statistic can be defined as follows:

$$X(s, k, \ell_0, \ell_1) = \frac{1}{\sqrt{n}} \sum_{(i,j) \in a(s,k,\ell_0,\ell_1)} x(i, j).$$

When this digital axoid does not overlap with the underlying feature, i.e., $a(s, k, \ell_0, \ell_1) \cap F = \emptyset$, one can easily derive

$$X(s, k, \ell_0, \ell_1) \sim N(0, 1).$$

When there is an intersection ($|a(s, k, \ell_0, \ell_1) \cap F| > 0$, where $|a(s, k, \ell_0, \ell_1) \cap F|$ is the number of 'common' pixels), we have

$$X(s, k, \ell_0, \ell_1) \sim N\left(\frac{1}{\sqrt{n}} |a(s, k, \ell_0, \ell_1) \cap F|, 1\right).$$

Notice that in the latter case, the value of the statistic $X(s, k, \ell_0, \ell_1)$ tends to be large.

A graph can be constructed. Let each digital axoid be a vertex. With a little bit abuse of the notation, we call it vertex $a(s, k, \ell_0, \ell_1)$. Two vertices are connected if and only if

1. they are at the same scale,

2. they are geometrically connected, and

3. they have similar slopes.

More details on the second point will be given in Section 5.3.2.4. Assign an edge between any pair of connected vertices. A graph is formed for all digital axoids.

Suppose that there is a prescribed constant N^*, such that a vertex is significant if and only if

$$X(s, k, \ell_0, \ell_1) > N^*.$$

A *significance graph* includes all the significant digital axoids and the edges connecting them. For a scale s, let $S(s)$ denote the significance graph at this scale.

An illustration of a significance graph is given in Figure 35, where three digital axoids are significant.

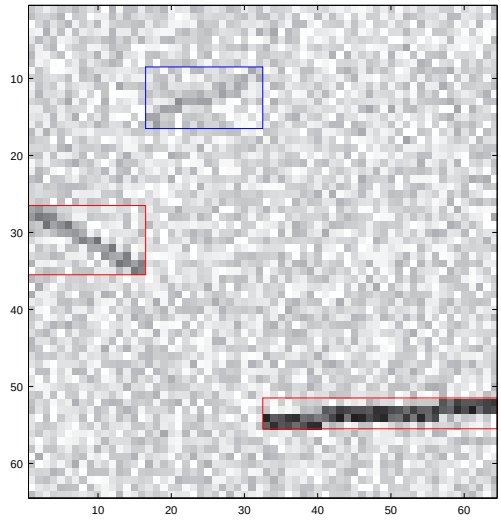


Figure 35: Illustration of a significance graph, having four vertices (i.e. significant digital axoids) and two are connected.

5.3.2.4 Good Continuation

We used the continuation (i.e. connectedness) in the graphic structure. Let ℓ_2 be another integer, $1 \leq \ell_2 \leq n$, the following two digital axoids are connected:

$$a(s, k, \ell_0, \ell_1), \quad a(s, k+1, \ell_0 + \ell_1, \ell_2 - (\ell_0 + \ell_1)).$$

This is a case of *continuation.*

The above condition will be modified in two ways.

1. First of all, the tail (i.e. the right side) of one digital axoid may not be the exact beginning (i.e. the left side) of another digital axoid.

2. The slopes of two digital axoids (which is controlled by ℓ_1) need to be close.

Later, one may observe that this modification will influence the type of the features that are detectable. Formally, for a prescribed constants K_1 and K_2, one has *good continuation* between axoid $a(s, k, \ell_0, \ell_1)$ and the following set of axoids:

$$a(s, k+1, \ell_0 + \ell_1 + c_1, \ell_1 + c_2),$$

where indices c_1 and c_2 satisfy the following conditions:

$$-K_1 \leq c_1 \leq K_1, \qquad 1 \leq \ell_0 + \ell_1 + c_1 \leq n, \qquad \text{and}$$
$$-K_2 \leq c_2 \leq K_2, \quad 1 \leq \ell_0 + 2\ell_1 + c_1 + c_2 \leq n.$$

The above conditions will guarantee that the second axoid is meaningful.

Examples of axoids presenting (or not presenting) good (respectively, bad) continuation are given in Figure 36 (respectively, 37).

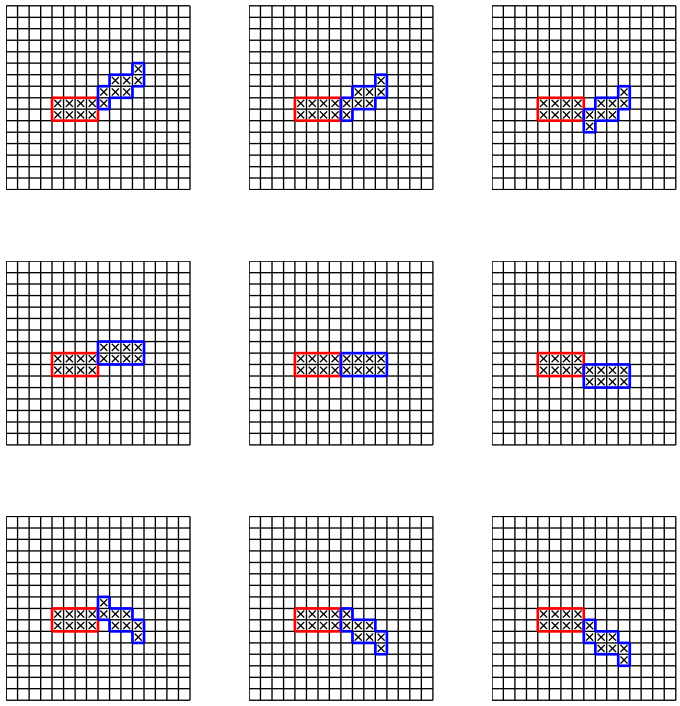


Figure 36: Cases of *good continuation*. Each row has the same amount of change of slopes. Each column has the same amount of vertical shifting.

In a significance graph, a two axoids are connected by an edge if and only if the good continuation condition is satisfied.

5.3.2.5 Longest Run

For a fixed scale s, there will be a significance graph, which contains a longest chain of connected significant axoids. Such a chain of axoids is called a *longest run*. At scale s, let $R(s)$ denote a longest run. The number of axoids in $R(s)$ will be denoted by $|R(s)|$. An illustration of a significance graph and its longest run is given in Figure 38.

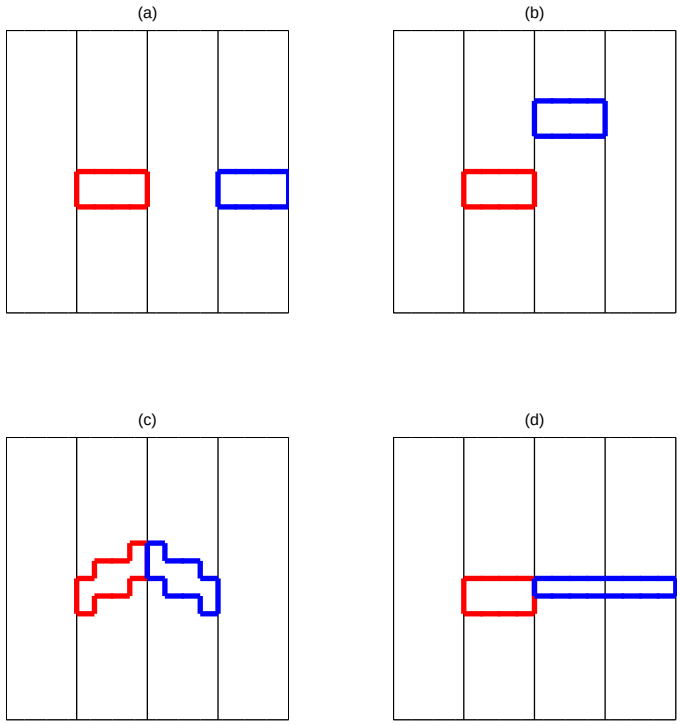


Figure 37: Cases of *bad continuation*: (a) and (b), separated axoids; (c) excessive change of slope; (d) not in the same scale.

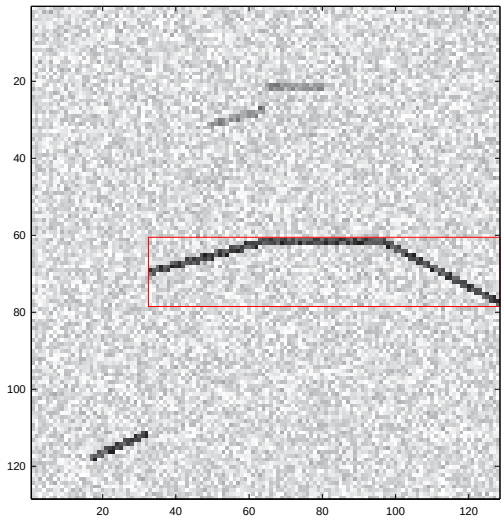


Figure 38: An example of a longest run. There are six vertices (i.e. significant digital axoids). The longest run contains three vertices.

5.3.3 Testing Presence of Filamentary Features

The following hypotheses testing problem is considered:

$$\begin{aligned} H_0: &\quad \forall (i,j), x(i,j) \sim N(0,1), \quad \text{and} \\ H_a: &\quad \exists F, A_n > 0, \text{ such that (5.3.41) is true.} \end{aligned}$$

When H_0 is true, the image is made by white noises. When H_a is true, there is an underlying feature embedded.

The testing procedure contains the following steps.

1. Compute $X(s,k,\ell_0,\ell_1)$ for all appropriate indices s,k,ℓ_0,ℓ_1.
2. Identify the significant graph $S(s)$ at each scale.
3. Find the longest path $R(s)$.
4. Compute the overall length of the longest paths:
$$L^o = \max_s |R(s)|.$$
5. Make a decision according to the following rules:
   - if $L^o > L^*$, where L^* is a prescribed number, then reject the null hypothesis H_0;
   - if $L^o \leq L^*$, accept the null hypothesis H_0.

By rejecting the null hypothesis, we conclude that there is sufficient evidence for the presence of an underlying feature; and vice versa.

Note that the above procedure is nearly parallel to the procedure in the paper [5].

5.3.3.1 Calibrate the N^*

Recall that in determining the significance graph, a parameter N^* is used (see Section 5.3.2.3). Recall that in [5], the key idea in determining the quantity N^* is to realize the following inequality:

$$P\{X(s,k,\ell_0,\ell_1) > N^*\} \leq \frac{p_0}{K_1 K_2},$$

where p_0 is a probability and constants K_1 and K_2 are introduced in Section 5.3.2.4. Since under the Null hypothesis, we have

$$X(s,k,\ell_0,\ell_1) \sim N(0,1).$$

One can derive the following,

$$N^* = \Phi^{-1}\left(1 - \frac{p_0}{K_1 K_2}\right), \tag{5.3.42}$$

where Φ^{-1} is the inverse of the cumulative distribution function of the standard normal distribution.

The above way of choosing quantity N^* has some theoretical optimality, as described in [5]. In simulations, a slightly smaller value for N^* may be more advantageous. Because it will lead to more significant axoids, and will allow detection of weaker underlying feature.

5.3.3.2 Calibrate the L^*

In theory, the quantity L^* is the maximum length of the longest run with a prescribed N^* and under the Null hypothesis. Such a quantity can be approximated by repeating the simulations (under H_0) for a large number of times, and take the maximum. Apparently this is a numerically expensive method.

We found in simulations that the length of the longest run closely follows a exponential distribution, with a fast dropping rate. Hence it is sensible to choose

$$L^* = C + L^*_{(0)},$$

where C is a constant (e.g., $C = 2$ or 3) and $L^*_{(0)}$ is the maximum length of the longest runs in a few (e.g., 10 times) simulations.

5.3.3.3 Detectability Threshold, $T^*(F)$

In theory, for a fixed feature F, there will be a constant $T^*(F)$ such that

- when $A_n > T^*(F)$, the underlying feature becomes detectable;
- when $A_n < T^*(F)$, the underlying feature is too fainted to be detected.

In our framework, we can design numerical experiments to identify such a threshold.

5.3.4 Simulations

5.3.4.1 Synthetic Data

Synthetic data are used in the simulations. We describe four types of geometric objects, which are representative on the filamentary features that are of interests.

1. *Trigonometric Functions.* Consider a function in a unit square:

$$y = 0.5 + \frac{(1-\delta_2)}{2}\sin[2\pi(1+\delta_1)(x-0.5)],$$

where $x \in (0.5 - 0.5/(1+\delta_1), 0.5 + 0.5/(1+\delta_1))$, and δ_1 and δ_2 are two constants: e.g., $\delta_1 = 0.5$ and $\delta_2 = 0.2$. A pixel (i, j) intersects with the above function if and only if the square $[(i-1)/n, i/n] \times [(j-1)/n, j/n]$ intersects with the above function. A feature F is made by all such pixels. An illustration of such a feature is given in Figure 39 (a).

2. *Thickened Trigonometric Functions.* The above feature can be 'thickened' by gluing the pixels immediately above or below into the feature. An illustration is given in Figure 39 (b).

3. *Beams.* We consider the function in a unit square:

$$y = 0.5 + a(x - 0.5),$$

where $x \in (0.5-0.5\delta_3, 0.5+0.5\delta_3)$. The following parameters may be chosen: $\delta_3 = 0.8$, $a = 1, 1/2, 1/4, ...$ The feature F is made by pixels that intersect the above function. An illustration is given in Figure 39 (c).

4. *Thickened Beams.* Again by gluing pixels immediately above and below a digital beam, we have a thickened beam. Let t denote the thickness (i.e., the number of pixels vertically aligned in a flat portion of the underlying feature). We may choose $t = 2, 4, 8$, etc.An illustration is given in Figure 39 (d).

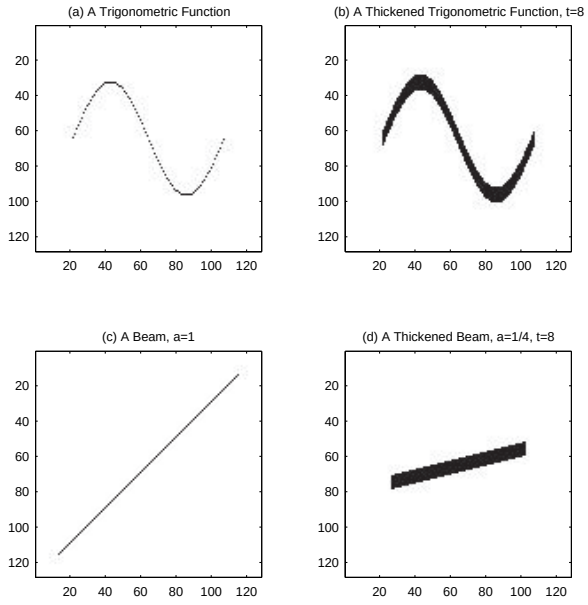


Figure 39: Features that are used in the numerical experiments.

5.3.4.2 Simulations

For a fixed value n, the threshold N^* is given by (5.3.42) (where we choose $p_0 = 1/2$, $K_1 = 2^s$, $K_2 = 2 \cdot S \cdot n/2^s$, and $S = 4$ is the maximum slope). The Table 7 gives a list of values of N^* that are used.

The threshold L^* is determined by simulations: $L^* = 2 + L^*_{(0)}$, where the quantity $L^*_{(0)}$ is the maximum length of the longest runs in ten simulations. The Table 8 gives the values of $L^*_{(0)}$ and L^* for a range of n's.

Table 7: Values of N^*.

$n =$	32	64	128	256	512	1024
$N^* =$	2.8856	3.0973	3.2972	3.4871	3.6683	3.8419

Table 8: Determining the values of L^*.

$n =$	32	64	128	256
$L^*_{(0)} =$	2	2	4	2
$L^* =$	4	4	6	4

The computing times for simulations in MATLAB at various sample sizes are given in Table 9.

Table 9: Computing time (in seconds) for each simulation.

$n =$	32	64	128	256
Computing Time	2.87	20.54	154.36	1264.8

For a given feature, e.g. a trigonometric function, one would like to know how large should the value of A_n be such that the underlying feature F can be reliably detected. For a fixed F (which is the first synthetic signal in Figure 39) and a fixed $n = 64$, applying N^* and $L^*(= 4)$ that are given above, Table 10 gives the numbers of detection out of 10 simulations. Figure 40 presents the noisy images with various amplitudes (A_n's).

5.3.4.3 More Observations

Increased sample size for the trigonometric features.

In another set of the experiments, the sample size is increased to 128×128. For different amplitudes A_n, the number of detections among 10 simulations are reported in Table 11. It is shown that when amplitude increased from 1/4 to 1/2, the underlying feature becomes detectable. Figure 41 presents the random images with the first three amplitudes that are used in Table 11. Notice that when $A_n = 0.5$, the underlying feature is still obscure, but the detection rate is nearly 100%.

Thinner objects. For an underlying trigonometric curve, when the sample size is still $n = 128$, however the thickness is reduced to $t = 1$, Table 12 presents the detection rates with a range of A_n's. Figure 42 presents some boundary cases.

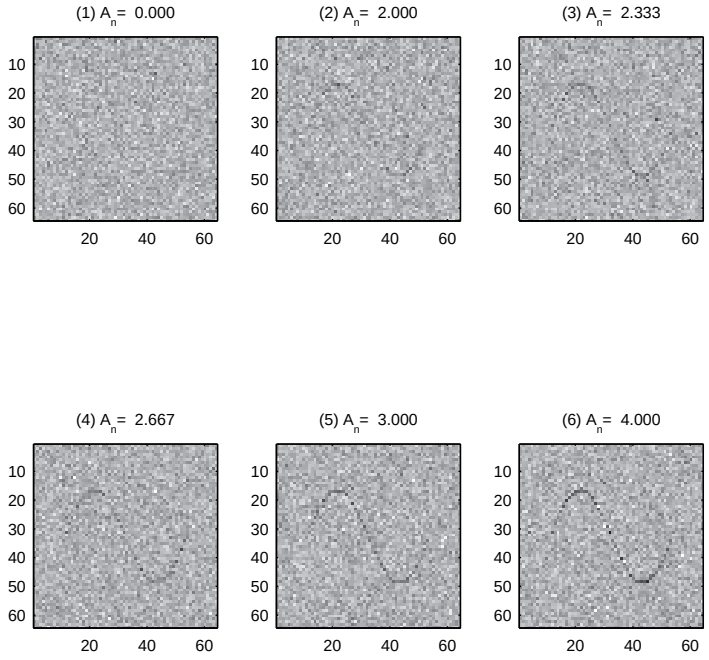


Figure 40: Noisy images with an underlying trigonometric curve having a range of amplitudes (provided in the titles). It shows that the proposed method can detect an underlying feature which is not obviously visible: case (5). Refer to Table 10.

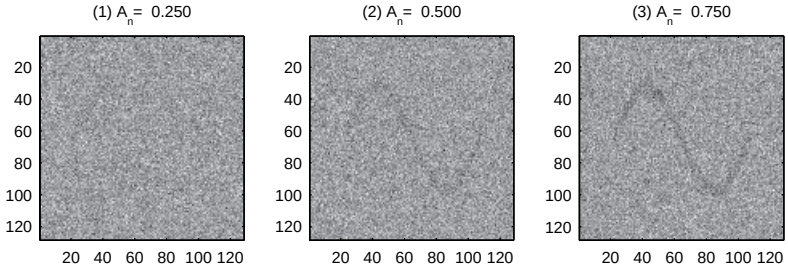


Figure 41: The random images that are used in the experiments that are reported in Table 11. The thickness of the underlying feature is $t = 8$.

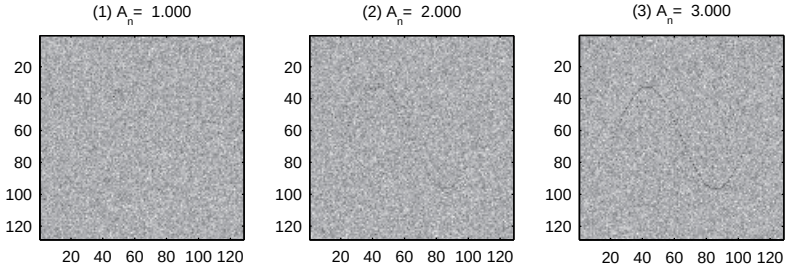


Figure 42: The random images that are used in the experiments that are reported in Table 12. The thickness of the underlying feature is reduced to $t = 1$.

Table 10: Detections out of 10 simulations for a trigonometric curve while $n = 64$ and $t = 1$.

$A_n =$	0	2.000	2.333	2.666	3.000	4.000
Detected Cases	0	0	4	7	9	10

Table 11: Detections out of 10 simulations for a trigonometric curve while $n = 128$ and $t = 8$.

$A_n =$	1/4	1/2	3/4	1
Detected Cases	0	10	10	10

5.3.4.4 Longest Runs

A key motivation of the proposed method is that the lengths of the longest runs are significantly larger when there are underlying features. Figure 43 illustrate this by presenting two comparison cases.

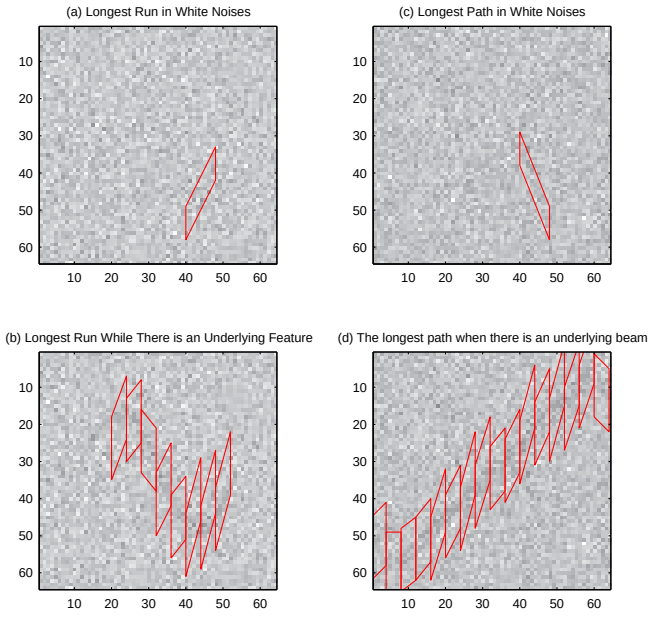


Figure 43: Illustration of an increment of length of the longest run when an underlying feature (a trigonometric function for (a) and (b), and a beam for (c) and (d)) is present. When there is an underlying feature, the length of the longest run is significantly larger.

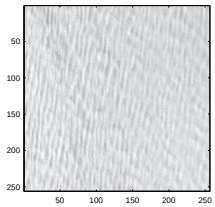
Table 12: Detections out of 10 simulations for a trigonometric curve while $n = 128$ and $t = 1$.

$A_n =$	1	2	3	4
Detected Cases	0	3	10	10

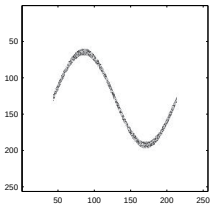
5.3.5 Conclusion

We present a method which is theoretically optimal and practically capable of detecting curvilinear features with arbitrary positions, orientations, anisotropic ratios, and smoothness. The proposed framework works for digital imageries. Simulations demonstrate the effectiveness of the algorithms. The future works include examining the variations in designing the digital axoids, and their impact on the efficiency and detection sensitivity of the proposed methodology.

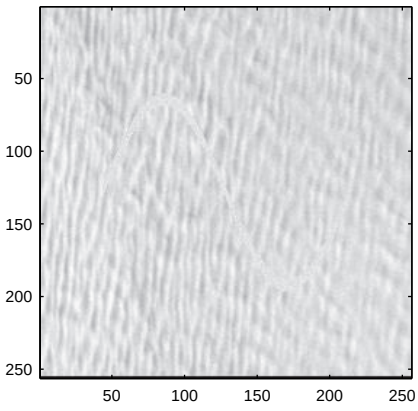
5.4 Detection in Textured Background



(a) Texture: water waves.



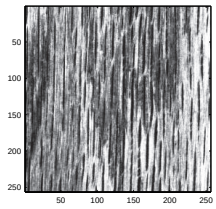
(b) Feature.



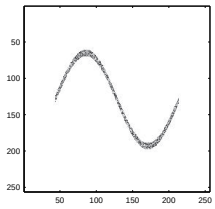
(c) Combined.

Figure 44: Example of a object (shaped like a trigonometric function, with its own textural distribution, as depicted in figure (b)) that is embedded in a textural image (figure (a)). Figure (c) is the combination: (c)=(a)+(b).

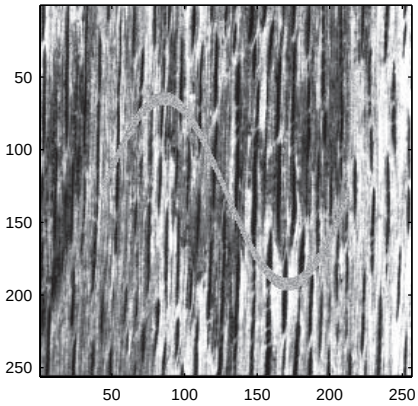
We consider detecting objects in a homogeneous background. The *objects* are regions within which the distributional properties of these image pixels are different from those



(a) Texture: wood waves.



(b) Object.



(c) Combined.

Figure 45: Another example of embedded object.

in the rest of the image. Two exemplary cases are given in Figure 44 and 45; in each case, there is a textural image, a trigonometric-function-shaped slim regions with contents different from the texture, and a combination of both of them. The detection problem is (1) to determine the presence of a object region, and furthermore (2) to infer the location and shape of the object region.

In this section, we explore the following idea: (1) the background makes the majority of an image, while a object region is the 'minority'; (2) In addition, the majority of the images (from the homogeneous background), if appropriately sampled, are located in a low-dimensional manifold; (3) For samples that overlap with the embedded region, they are 'away' from the manifold. Given that the above three conjectures are true, the distance from a sampled patch to the underlying manifold gives the probability that a sample overlaps with the embedded object. If all the *high probability samples* are relative concentrated, then one has evidence for the presence of an embedded object; otherwise there may not be an embedded object. An illustration of an underlying manifold for samples (e.g., patches) from a homogeneous background is given in Figure 46. The *significantly run algorithms* [5, 43, 42] can be used to process the patterns of the high probability samples. The distance from a sample to an underlying manifold can be empirically estimated by an algorithm—Local Linear Projection (LLP)—that is designed in other occasions [41, 44]. The principal idea of LLP is inspired by LLE [56] and ISOMAP [62]. Simulations demonstrated the effectiveness of this approach.

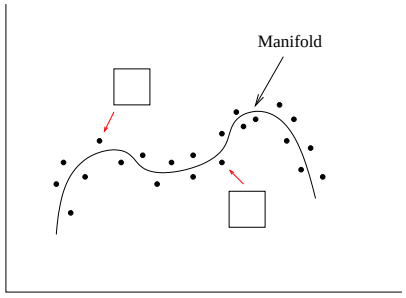


Figure 46: An illustration of the presence of a manifold.

5.4.1 Formulation

For an N by N image, let $y_i, i \in \mathcal{I}$, denote all of the 8 by 8 sampled patches with two diagonal corners being at $(4a+1, 4b+1)$ and $(4a+8, 4b+8)$, where $0 \leq a, b \leq (N-8)/4$. The patch size (8×8) is chosen for computational convenience. We assume that if patch y_i is sampled in the background, then

$$y_i = f(t_i) + \varepsilon_i, \quad i \in \mathcal{I},$$

where function $f(\cdot)$ is a locally smooth function that determines the underlying manifold, t_i's denote the underlying parameters for the manifold, and ε_i's are random errors.

5.4.2 Distance to Manifold

For any patch, y_i, the distance from this patch to its original image on the manifold, which is $f(t_i)$, is

$$\|y_i - f(t_i)\|_2. \tag{5.4.43}$$

As explained earlier, this distance measures how likely the patch is in the background. The larger the above distance is, the less likely this patch is on the background. An illustration

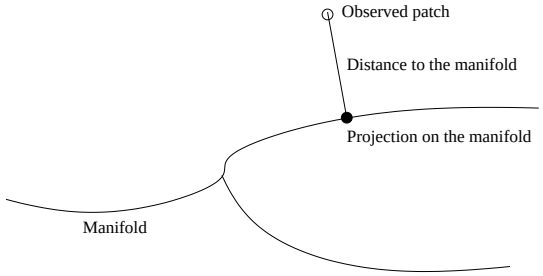


Figure 47: An illustration of distance from an observed patch to the manifold.

of the distance from a patch to the manifold is given in Figure 47. Note that function $f(\cdot)$ is not available. In the next section, a numerical method is designed to estimate this distance.

5.4.3 LLP: Local Linear Projection

An LLP can be applied to extract the local low-dimensional structure. In the first step, neighboring observations are identified. In the second step, a Singular Value Decomposition (SVD) or a Principal Component Analysis (PCA) is used to estimate the local linear subspace. Finally, the observation is projected into this subspace. An illustration of LLP in 2-D with local dimension being equal to 1 (i.e., linear) and 15 nearest neighbors is provided in Figure 48. A detailed description of the algorithm is given in the following.

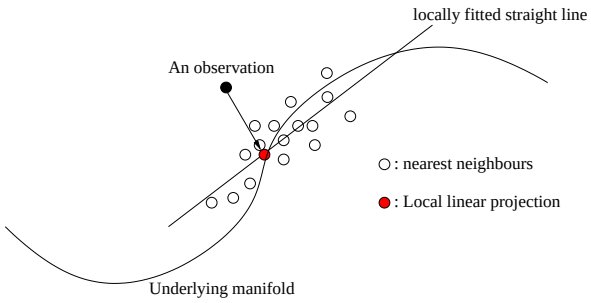


Figure 48: An illustration of Local Linear Projection in a 2-D space with local dimension being equal to 1 and 15 nearest neighbors.

ALGORITHM: LLP

for each observation $y_i, i = 1, 2, 3, \ldots, N$,

1. Find the K-nearest neighbors of y_i. The neighboring points are denoted by $\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_K$.

2. Use PCA or SVD to identify the linear subspace that contains most of the information on vectors $\tilde{y}_1$, $\tilde{y}_2$, ..., $\tilde{y}_K$. Suppose the linear subspace is $\mathcal{A}_i$, and $P_{\mathcal{A}_i}(x)$ denote the projection of a vector x into this subspace. Let k_0 denote the assumed dimension of the embedded manifold, then subspace $\mathcal{A}_i$ can be viewed as a linear subspace spanned by the vectors associated with the first k_0 singular values.

3. Project y_i into the linear subspace $\mathcal{A}_i$ and let $\hat{y}_i$ denote this projection: $\hat{y}_i = P_{\mathcal{A}_i}(y_i)$.

end.

The distance between y_i ($i \in \mathcal{I}$) and $f(t_i)$ can be estimated by $\|y_i - \hat{y}_i\|_2$.

5.4.4 SRA: Significant Run Algorithm

Even though the distance to a manifold can be estimated, it still remains unclear when the distance is *significantly* large. Instead of studying the distribution of the distances

themselves, we study their spatial patterns by using a *significance run algorithm* (SRA), which was introduced in [5], and was later used in [43] and [42].

A summary of a SRA is as follows. Each patch is associated with a node. Because patches are equally spaced, they form a table like in Figure 49. There is an edge between

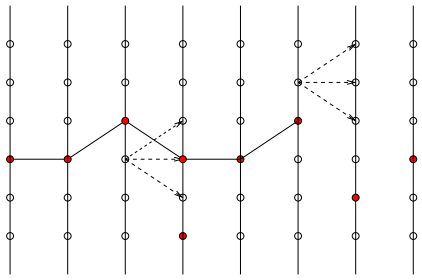


Figure 49: An illustration of Significance Graph and a Significance Run.

two nodes iff their corresponding patches are spatially connected. A node is significant iff the corresponding distance $\|y_i - \hat{y}_i\|_2$ is above a prescribed threshold (denoted by $\mathcal{T}_1$). A *significance run* is a chain of connected significant nodes. The length of the longest significance run is the test statistic: an embedded object is claimed to be present iff this length is above a constant (denoted by $\mathcal{T}_2$). It has been shown in other occasions (e.g., [5, 42]) that SRA leads to a powerful test.

Note that both $\mathcal{T}_1$ and $\mathcal{T}_2$ can be determined numerically. Constant $\mathcal{T}_1$ can be a given percentile of the empirical estimates of the distances: $\|y_i - \hat{y}_i\|_2$. Constant $\mathcal{T}_1$ can be derived from simulations.

5.4.5 Parameter Estimation

In the LLP, one needs to specify the number of nearest neighbors and the local dimension. This can be done by studying the empirical distribution of the distances and the total residual sums of squares.

5.4.5.1 Number of Nearest Neighbors

An illustration of the percentiles of the distances to the nearest neighbors are given in Figure 50. We choose 50 nearest neighbors, because it is approximately a kink point in this figure. It is possible to choose the number of nearest neighbors by studying the distances to the nearest neighbors. In this section, we do not pursuit in this direction.

5.4.5.2 Local Dimension

The problem of estimating local dimensionality has been analyzed in [56] and [62]. There are followup works in this line. Due to space, we omit the details. Figure 51 gives the plot of the residual sums of squares ($\sum_{i\in\mathcal{I}} \|y_i - \hat{y}_i\|_2^2$) versus the local dimensions (as k_0 in the

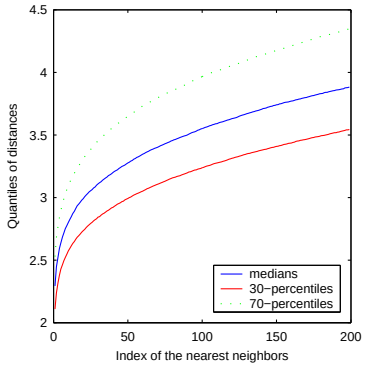


Figure 50: Percentiles of the distances from the nearest neighbors.

LLP). An approximate kink point is at $k_0 = 15$, which is our choice of local dimension in

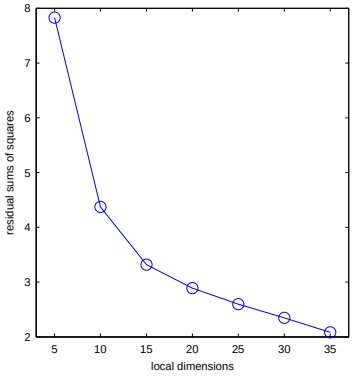


Figure 51: Residual sum of squares versus local dimensions.

Simulations.

5.4.6 Simulations

We apply the above approach to the two figures in Figure 44 (c) and Figure 45 (c). The positions of the significant patches are displayed in Figure 52 (for the water image) and in Figure 53 (for the wood image) respectively. In both cases, the constant $\mathcal{T}_1$ is chosen to be the 95th percentile of the squared distances: $\|y_i - \hat{y}_i\|_2^2, \forall i \in \mathcal{I}$. Obviously, the significant patches are concentrated around the embedded object, which is the trigonometric shape. Hence a SRA will unveil the presence of the object.

For comparison, Figure 54 gives the patterns of significant patches when there is no embedded objects.

5.4.7 Conclusions

A framework that utilizes the underlying geometric distribution is proposed to detect objects in a homogeneous background. Simulations showed the promises of the proposed method.

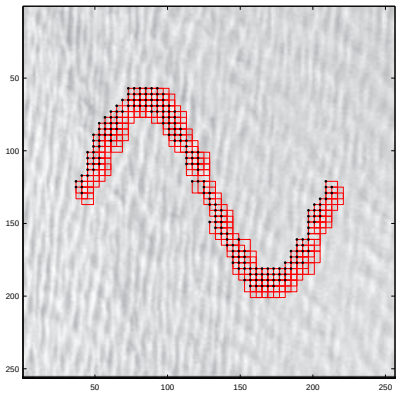


Figure 52: Pattern of significant patches for water image. Northwestern corners of significant patches are marked by dark dots.

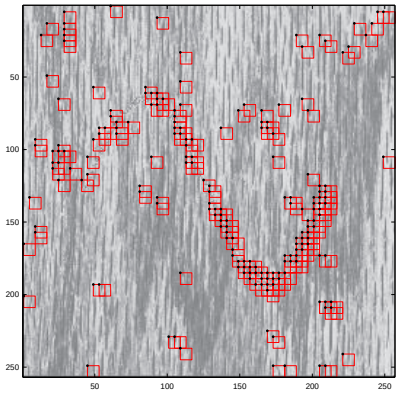


Figure 53: Pattern of significant patches for wood image. Northwestern corners of significant patches are again marked by dark dots.

The proposed method takes advantages of an underlying manifold, on which the sampled patches from the image reside. Significant Run Algorithm is utilized to generate a hypothesis testing procedure.

By modifying the structure of the significant graph, the above approach can be generalized for more general object, e.g., instead of functions, one can consider curves, or even non-filamentary objects. We leave this as a future work.

If the background is non-homogeneous, which is true in many cases, then the above approach will fail. The proposed framework can be utilized to derive a general theory on when an embedded object is detectable, and when it is not. This will be another future work.

5.5 The Longest Significance Run in a Bernoulli Net

In previous sections, by applying the Significant Run Algorithm, the detection problems are formulated as hypothesis testing problems. This section studies the properties of the null

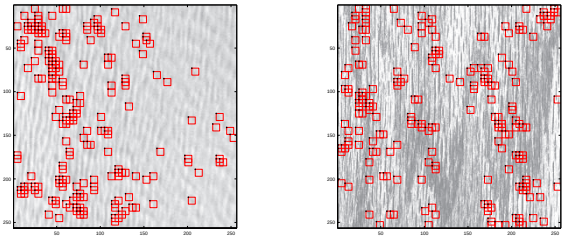


Figure 54: Pattern of significant patches for water and wood image, while there is *no* embedded object.

distribution and the parametrical selection such that the power of the test can be optimized.

5.5.1 Introduction

Over the years, there has been considerable research work regarding the length of the longest success run L_n in n Bernoulli trials whose extensive applications include signal detection, reliability, quality control, radar astronomy, DNA sequence analysis, and star-up demonstration testing, etc. Various expressions for the exact distribution of L_n were given in [15], [33] and elsewhere. Nevertheless, the complexity of these combinatoric formulae may bring difficulty in computation and prevent its direct use in the analysis of theoretical properties. Several asymptotic and approximate formula and bounds have been developed to overcome this problem. For example, one of the most famous asymptotic results was given in a series of papers by [54], [24] and [25], which shows that, as $n \to \infty$,

$$\frac{L_n}{\log_{1/\rho} n} \longrightarrow 1 \text{ almost surely.} \tag{5.5.44}$$

Interested readers may refer to the book by [9] for up-to-date reviews on the developments in related problems.

Although many methods have been developed to deal with one dimensional longest run statistics, the work in two or higher dimensional analogues is comparatively small and has been focused on scanning for unusual clusters in a small subregion. For example, a multidimensional analogue concerning the largest cube of 1's in random lattices was investigated in [19].

The problem that interests us is the detection of unusual curvilinear structure in a two dimensional lattice system. Two nodes are defined as connected if they may be neighbors on the curves of interest. The length of the longest connected nodes of 1's, which we shall call the longest significance run, is used as the test statistics. If the length exceeds a certain threshold, we shall say that there exists a curve with different distributions from other majority nodes. This problem is motivated by a recent paper [5].

The purpose of this section is to develop theoretical analysis on the distribution of the longest significant run and the parametrical selection for the optimization of the power.

We next describe the formal formulation of this problem. Let $[1,n] \times [1,m]$ be a two dimensional rectangular region. Assume on each node (i,j), a Bernoulli variable X_{ij} is observed. If $X_{i,j} = 1$, the node is called *significant*. Any two nodes (i_1, j_1) and (i_2, j_2) are defined as *connected* if $|i_1 - i_2| = 1$ and $|j_1 - j_2| \leq C$, with C being a prescribed constant. Define a *filament* of length ℓ as a chain of ℓ *connected* nodes:

$$R(\ell) = \{(i_1 + 1, j_2), \ldots, (i_1 + \ell - 1, j_\ell) : |j_k - j_{k-1}| \leq C, 2 \leq k \leq \ell\}.$$

A *significant run* refers to a filament with all nodes significant. We call the system a *Bernoulli net*, and we are interested in testing the null hypothesis of a constant success probability p against the alternative that some filament with unknown location and length has a higher success probability. Define L_0 the length of the longest significant run on the Bernoulli net. If L_0 exceeds a preassigned threshold, we will claim the existence of a filamentary structure.

5.5.2 The Distribution of the Longest Significance Run

For the simple Bernoulli net with $C = 0$, we have

$$P(L_0 \geq \ell | n, m, p) = 1 - P(L_0 < \ell | n, m, p) = 1 - [1 - P(L_0 \geq \ell | n, 1, p)]^m,$$

where $P(L_0 \geq \ell | n, 1, p)$ is the distribution of the longest run in a one dimensional Bernoulli sequence.

In the following, some of the approaches and techniques for handling one-dimensional longest run are extended to handle our two-dimensional analogue.

5.5.2.1 Bounds

For one dimensional case ($m = 1$), the following simple bound was originally developed in reliability-focused works [53]

$$(1 - p^\ell)^{n-\ell+1} \leq P(L_n < \ell) \leq (1 - qp^\ell)^{n-\ell+1},$$

where $q = 1 - p$. Denote $P_\ell = P(L_0 = \ell | \ell, m, C, p)$ the probability that the length of the longest run is ℓ when there are exact ℓ columns. A direct extension of the above bounds to a two dimensional Bernoulli net yields the following theorem.

Theorem 5.

$$(1 - P_\ell)^{n-\ell+1} \leq P(L_0 < \ell | n, m, C) \leq [1 - q^m P_\ell]^{n-\ell+1}. \tag{5.5.45}$$

Proof. We introduce the following notations.

E_i : event that the longest run is shorter than ℓ in the subregion $[i, i + \ell - 1] \times [1, m], 1 \leq i \leq n - \ell + 1$.

F_i : event that the longest run is shorter than ℓ in the subregion $[1, i] \times [1, m], \ell \leq i \leq n$.

F_i' : complement of F_i.

G_i : event that there is no significant node on the $(i - \ell)$th column.

Lower bound:

$$P(L_0 < \ell | n, m) = P(E_1 \cap E_2 \cap \cdots \cap E_{n-\ell+1}) \geq \prod_{i=1}^{n-\ell+1} P(E_i) = (1 - P_\ell)^{n-\ell+1}.$$

Upper bound:

$$\begin{aligned} P(L_0 < \ell | n, m) &= P(F_n) = P(F_\ell) \prod_{i=\ell+1}^{n} P(F_i)/P(F_{i-1}) \\ &= P(F_\ell) \prod_{i=\ell+1}^{n} [1 - P(F_i'|F_{i-1})]. \end{aligned}$$

$$\begin{aligned} P(F_i'|F_{i-1}) &\geq P(F_i'|G_i F_{i-1}) \cdot P(G_i|F_{i-1}) \\ &\geq P_\ell \cdot P(G_i) \\ &= (1-p)^m P_\ell \end{aligned}$$

The last inequality holds is because $P(G_i|F_{i-1}) \geq P(G_i|F_{i-1}')$, therefore

$$\begin{aligned} P(G_i) &= P(G_i|F_{i-1})P(F_{i-1}) + P(G_i|F_{i-1}')P(F_{i-1}') \\ &= P(G_i|F_{i-1}) - [P(G_i|F_{i-1}) - P(G_i|F_{i-1}')]P(F_{i-1}') \\ &\leq P(G_i|F_{i-1}) \end{aligned}$$

□

These bounds are very loose especially when m is large. However, the meaning of the bounds is to derive next asymptotic results.

5.5.2.2 Asymptotic

Let

$$\rho_\ell = P_\ell / P_{\ell-1}. \tag{5.5.46}$$

The ratio means the conditional probability that the length of the longest run is ℓ given that there exits longest runs with length $\ell - 1$ in previous $(\ell - 1)$ columns.

Lemma 4. *There exists a constant ρ depending only on p, C, and m, such that*

$$\lim_{\ell \to \infty} \rho_\ell = \rho, \tag{5.5.47}$$

Proof. Let $\{\mathbf{x}_i, i = 1, 2, \ldots\}$ be a Markov Chain, where $\mathbf{x}_i = (x_{i,1}, \cdots, x_{i,m})$ is a vector of length m. $x_{i,j}$ denotes the state of node (i, j). $x_{ij} = 1$ if the length of the longest run up to node (i, j) is equal to i, otherwise $x_{ij} = 0$. Let S be the set of all the possible values of $\mathbf{x}_i$. The cardinality of S is 2^m.

For $j = 1, \cdots, m$, denote $\Omega(j) = \{j' : |j' - j| \leq C, 1 \leq j' \leq m\}$ the set containing neighboring indexes of j. For two states $\mathbf{s}_1, \mathbf{s}_2 \in S$, the transition probability is

$$P_{\mathbf{s}_1\mathbf{s}_2} = p^a(1-p)^b\mathbf{1}(c = 0), \tag{5.5.48}$$

where

$$a = \sum_{j=1}^{m} \mathbf{1}(\max_{j' \in \Omega(j)} \mathbf{s}_1(j') = 1, \mathbf{s}_2(j) = 1),$$

$$b = \sum_{j} \mathbf{1}(\max_{j' \in \Omega(j)} \mathbf{s}_1(j') = 1, \mathbf{s}_2(j) = 0),$$

$$c = \sum_{j} \mathbf{1}(\max_{j' \in \Omega(j)} \mathbf{s}_1(j') = 0, \mathbf{s}_2(j) = 1).$$

Obviously, $c = 0$ When $\mathbf{s}_1 = \mathbf{s}_2$. Therefore $P_{\mathbf{ss}} > 0 \ \forall \mathbf{s} \in S$.

Let $\pi_{\mathbf{s}}^i = P\{\mathbf{x}_i = \mathbf{s} | \mathbf{x}_i \neq \mathbf{0}\}$, we have

$$\rho_\ell = \frac{P_\ell}{P_{\ell-1}} = \frac{P\{\mathbf{x}_l \neq \mathbf{0}\}}{P\{\mathbf{x}_{l-1} \neq \mathbf{0}\}} = P\{\mathbf{x}_l \neq \mathbf{0} | \mathbf{x}_{l-1} \neq \mathbf{0}\} = \sum_{\mathbf{s} \neq \mathbf{0}} \pi_{\mathbf{s}}^{\ell-1}(1 - P_{\mathbf{s0}}).$$

Now define another Markov Chain $\mathbf{y}_i = \{\mathbf{x}_i | \mathbf{x}_i \neq 0\}$, with state space $S/\{\mathbf{0}\}$. The transition probability of the new Markov chain is

$$P'_{\mathbf{s}_1\mathbf{s}_2} = \frac{P_{\mathbf{s}_1\mathbf{s}_2}}{\sum_{\mathbf{s} \neq \mathbf{0}} P_{\mathbf{s}_1\mathbf{s}}}, \quad \mathbf{s}_1, \mathbf{s}_2 \in S/\{\mathbf{0}\}.$$

It is obvious that $P'_{\mathbf{ss}} > P_{\mathbf{ss}} > 0$. Therefore the Markov chain is aperiodic.

Moreover, it is easy to see that any other state is accessible from the state $(1, 1, \cdots, 1)$ in one step. Also, the state $(1, 1, \cdots, 1)$ is accessible from any other state in m steps. So all states are communicated with each other. Therefore the new Markov chain is irreducible.

Because the Markov chain $\{\mathbf{y}_i, i = 1, 2, \cdots\}$ is finite, aperiodic, and irreducible, there exists limiting distribution $\pi_{\mathbf{s}}$ such that

$$\lim_{\ell \to \infty} \pi_{\mathbf{s}}^\ell = \pi_{\mathbf{s}}.$$

Therefore,

$$\lim_{\ell \to \infty} \rho_\ell = \sum_{\mathbf{s} \neq \mathbf{0}} \pi_{\mathbf{s}} \cdot (1 - P_{\mathbf{s0}}) = \rho. \tag{5.5.49}$$

□

In the following, by allowing we use L_n to denote the longest run in a Bernoulli net.

Theorem 6. *As $n \to \infty$,*

$$\frac{L_n}{\log_{1/\rho} n} \longrightarrow 1 \text{ almost surely.} \tag{5.5.50}$$

Proof. For any real value x, we define

$$L_x = L_{\lfloor x \rfloor}.$$

First, we will prove that

$$\frac{L_{e^k}}{\log_{1/\rho} e^k} \longrightarrow 1 \text{ as } k \to \infty \text{ almost surely.} \tag{5.5.51}$$

To prove (5.5.51), we need to prove that $\forall \epsilon > 0$,

$$P(|\frac{L_{e^k}}{\log_{1/\rho} e^k} - 1| > \epsilon \text{ infinity often}) = 0.$$

It is sufficient to prove that

$$\sum_k P(|\frac{L_{e^k}}{\log_{1/\rho} e^k} - 1| > \epsilon) < \infty. \tag{5.5.52}$$

From (5.5.47), $\forall \delta > 0$, $\exists \ell_0$ such that when $\ell > \ell_0$,

$$\rho^{1+\delta} \leq \frac{P_\ell}{P_{\ell-1}} \leq \rho^{1-\delta}.$$

Therefore

$$P_{\ell_0} \rho^{(1+\delta)(\ell-\ell_0)} \leq P_\ell \leq P_{\ell_0} \rho^{(1-\delta)(\ell-\ell_0)}.$$

From (5.5.45), we have

$$[1 - a_1 \rho^{\ell(1-\delta)}]^{\lfloor e^k \rfloor - \ell + 1} \leq P(L_{e^k} < \ell) \leq [1 - a_2 \rho^{\ell(1+\delta)}]^{\lfloor e^k \rfloor - \ell + 1},$$

where $a_1 = P_{\ell_0} \rho^{-\ell_0(1+\delta)}, a_2 = (1-p)^m P_{\ell_0} \rho^{-\ell_0(1-\delta)}$.

$$\begin{aligned}
&P(|\frac{L_{e^k}}{\log_{1/\rho} e^k} - 1| > \epsilon) \\
&= P(L_{e^k} > (1+\epsilon) \log_{1/\rho} e^k) + P(L_{e^k} < (1-\epsilon) \log_{1/\rho} e^k) \\
&\leq 1 - \{1 - a_1 \rho^{(1-\delta)\lfloor (1+\epsilon) \log_{1/\rho} e^k \rfloor}\}^{\lfloor e^k \rfloor - \lfloor (1+\epsilon) \log_{1/\rho} e^k \rfloor + 1} \\
&\quad + \{1 - a_2 \rho^{(1+\delta)\lceil (1-\epsilon) \log_{1/\rho} e^k \rceil}\}^{\lfloor e^k \rfloor - \lceil (1-\epsilon) \log_{1/\rho} e^k \rceil + 1} \\
&\leq 1 - \{1 - a_1 \rho^{(1-\delta)[(1+\epsilon) \log_{1/\rho} e^k - 1]}\}^{e^k - (1+\epsilon) \log_{1/\rho} e^k + 2} \\
&\quad + \{1 - a_2 \rho^{(1+\delta)[(1-\epsilon) \log_{1/\rho} e^k + 1]}\}^{e^k - (1-\epsilon) \log_{1/\rho} e^k - 1}
\end{aligned} \tag{5.5.53}$$

$\exists k_0$, such that when $k > k_0$,

$$(1+\epsilon) \log_{1/\rho} e^k \geq 2, \text{ and } (1-\epsilon) \log_{1/\rho} e^k + 1 \leq e^k/2. \tag{5.5.54}$$

Choose $\delta = \frac{1}{4}\epsilon$, then

$$(1-\delta)(1+\epsilon) \geq 1 + \frac{\epsilon}{2} \text{ and } (1+\delta)(1-\epsilon) \leq 1 - \frac{\epsilon}{2}. \tag{5.5.55}$$

Substitute (5.5.54) and (5.5.55) into (5.5.53),

$$P(|\frac{L_{e^k}}{\log_{1/\rho} e^k} - 1| > \epsilon) \leq 1 - [1 - a_1' e^{-(1+\epsilon/2)k}]^{e^k} + [1 - a_2' e^{-(1-\epsilon/2)k}]^{e^k/2}, \tag{5.5.56}$$

where $a_1' = a_1 \rho^{-(1-\delta)}$ and $a_2' = a_2 \rho^{(1+\delta)}$. Since

$$\lim_{k\to\infty} [1 - a_1' e^{-(1+\epsilon/2)k}]^{e^{(1+\epsilon/2)k/a_1'}} = e^{-1},$$

For $\delta_1 = e^{-1}/2,\ \exists k_1$, when $k > k_1$,

$$[1 - a_1' e^{-(1+\epsilon/2)k}]^{e^{(1+\epsilon/2)k/a_1'}} \geq e^{-1} - \delta_1,$$

and

$$[1 - a_2' e^{-(1-\epsilon/2)k}]^{e^{(1-\epsilon/2)k/a_2'}} \leq e^{-1} + \delta_1.$$

Let $b_1 = e^{-1}/2$, and $b_2 = 3e^{-1}/2$. Then for $k > k_2 = \max(k_0, k_1)$, we have

$$P(|\frac{L_{e^k}}{\log_{1/\rho} e^k} - 1| > \epsilon) \leq 1 - b_1^{a_1' e^{-k\epsilon/2}} + b_2^{\frac{a_2'}{2} e^{k\epsilon/2}} \leq (a_1' \ln \frac{1}{b_1}) e^{-k\epsilon/2} + b_2^{\frac{a_2'}{4} k\epsilon}. \tag{5.5.57}$$

Note that $1 - b_1^x \leq -x \ln b_1$ and $e^x \geq x$. We have,

$$\sum_{k_2}^{\infty} P(|\frac{L_{e^k}}{\log_{1/\rho} e^k} - 1| > \epsilon) < \infty.$$

Therefore (5.5.51) holds as $k \to \infty$.

Denote

$$f_k = \frac{L_{e^k}}{\log_{1/\rho} e^k},$$

For any n, there is k_n such that $e^{k_n} \leq n \leq e^{k_n+1}$, since

$$f_{k_n} \frac{k_n}{k_n+1} \leq \frac{L_n}{\log_{1/\rho} n} \leq f_{k_n+1} \frac{k_n+1}{k_n},$$

we have

$$\frac{L_n}{\log_{1/\rho} n} \to 1 \text{ when } n \to \infty.$$

. □

5.5.2.3 Approximation

For large m or n, we derive the following approximation using an approach similar to [68].

We first consider the case when m is large. Let A_k denote the event that the longest run is shorter than ℓ in the subregion $[1,n] \times [(k-1)C\ell + 1, (k-1)C\ell + 2C\ell], 1 \le k \le r_1$, where we assume $r_1 = m/\ell - 1$ is an integer. We have

$$\begin{aligned} P(L_0 < \ell | n, m, C) &= P(A_1 A_2 \cdots A_{r_1}) \\ &\approx P(A_1) P(A_2|A_1) P(A_3|A_2) \cdots P(A_{r_1}|A_{r_1-1}). \end{aligned}$$

Moreover,

$$P(A_i|A_{i-1}) \approx \frac{Q(n, 2C\ell)}{Q(n, C\ell)},$$

where $Q(n, iC\ell) = P(L_0 < \ell | n, iC\ell), i = 1, 2$. Therefore,

$$P(L_0 \ge \ell | n, m) \approx 1 - Q(n, 2C\ell)[Q(n, 2C\ell)/Q(n, C\ell)]^{m/\ell - 2}. \tag{5.5.58}$$

Similarly, we can derive an approximation for large n. let B_k denote the event that the longest run is shorter than ℓ in the subregion $[(k-1)\ell+1, (k-1)\ell+2\ell] \times [1, m], 1 \le k \le r_2 - 1$, where $r_2 = n/\ell - 1$. We have

$$\begin{aligned} P(L_0 < \ell | n, m, C) &= P(B_1 B_2 \cdots B_{r_2}) \\ &\approx P(B_1) P(B_2|B_1) P(B_3|B_2) \cdots P(B_{r_2}|B_{r_2-1}). \end{aligned}$$

Again,

$$P(L_0 \ge \ell | n, m, C) \approx 1 - Q(2\ell, m)[Q(2\ell, m)/Q(\ell, m)]^{n/\ell - 2}, \tag{5.5.59}$$

where $Q(i\ell, m) = P(L_0 < \ell | i\ell, m), i = 1, 2$.

When both n and m are large, we combining (5.5.58) and (5.5.59),

$$P(L_0 \ge \ell | n, m, C) \approx 1 - Q_2(Q_2/Q_1)^{n/\ell - 2}, \tag{5.5.60}$$

where $Q_i = Q_{i1}(Q_{i2}/Q_{i1})^{m/\ell - 2}$ and $Q_{ij} = P(L_0 < \ell | i\ell, j\ell), i, j = 1, 2$.

5.5.2.4 A Simulation Approach

Denote $L_0(p)$ the length of the longest significance run for a given probability p. Below we provide a dynamic programming approach to compute $L_0(p)$ for the entire real line simultaneously.

Let $Y_{i,j} \sim \text{Uniform}(0,1), 1 \le i \le n, 1 \le j \le m$ be i.i.d uniform random variables. For a given probability p, node (i, j) is significant if $Y_{1,j} \ge 1 - p$. Let $L_{i,j}(p)$ denote the length of the longest significant run up to node (i, j).

For the nodes in the first column, $(1, j)$, $j = 1, 2, \ldots, m$, we have

$$L_{1,j}(p) = \begin{cases} 0, & \text{if } p < 1 - Y_{1,j}, \\ 1, & \text{otherwise.} \end{cases}$$

For node (i, j), it is not hard to verify the following

$$L_{i,j}(p) = \mathbf{1}\{Y_{i,j} \geq 1 - p\} \cdot \left\{ 1 + \max_{j' \in \Omega(j)} L_{i-1,j'}(p) \right\},$$

where, as defined before, $\Omega(j) = \{j' : |j' - j| \leq C, 1 \leq j' \leq m\}$ is the set containing neighboring indexes of j. Function $\mathbf{1}\{Y_{i,j} > 1 - p\}$ is an indicator function.

When all $L_{i,j}(p)$ are available, the value of $L_0(p)$ satisfies

$$L_0(p) = \max_{i,j} L_{i,j}(p).$$

The function $L_{i,j}(p)$ is piecewise constant, and an increasing function of p. Define the break points in function $L_{i,j}(p)$ as follows,

$$b_\ell^{(i,j)} = \min\{p : L_{i,j}(p) \geq \ell\}, \quad \ell = 1, 2, \ldots, n.$$

The value $b_\ell^{(i,j)}$ is the lower bound of set when the value of $L_{i,j}(\cdot)$ is equal to ℓ. For nodes in the first column, we have

$$b_1^{(1,j)} = 1 - Y_{1,j}, \forall j.$$

Since $L_{1,j}(\cdot) \leq 1$, we can assume

$$b_\ell^{(1,j)} = 1, \forall j \text{ and } \ell > 1.$$

We can derive the following updating scheme for the break points:

$$b_1^{(i,j)} = 1 - Y_{i,j},$$

and for $\ell \geq 2$,

$$b_\ell^{(i,j)} = \max\left(b_1^{(i,j)}, \min_{j' \in \Omega(j)} b_{\ell-1}^{(i-1,j')} \right).$$

Define

$$b_\ell^* = \min\{p : L_0(p) \geq \ell\}.$$

It is not hard to observe that

$$b_\ell^* = \min_{i,j} b_\ell^{(i,j)}, \quad \forall \ell = 1, 2, \ldots, n.$$

There are n break points. For each break point, there are at most $(2C + 1)$ previous break points to compare. Hence it takes at most $(2C + 1)n$ operations to compute break points for a new node. The time of the algorithm is at most $mn^2(2C + 1)$, and the space is $mn(n + 1)/2$.

For each simulation, the results of break points are arranged in a $1 \times n$ vector. By conducting N simulations, we obtain a matrix of size $N \times n$. Given p, the probability $P(L_0(p) \geq \ell)$ is estimated by the fraction of b_ℓ^*'s that are smaller than p.

5.5.2.5 Numerical Results

A. Asymptotic behavior

First, we study the dependence of ρ on m. The Markov Chain approach describe in the proof of Lemma 1 is employed, and ρ is calculated from (5.5.49). The algorithm has complexity $O(2^m)$. In the following figure, we plot the curve of ρ versus p for three different m's: $m = 4, 8, 10$. We observe that ρ does not increase much when m increases.

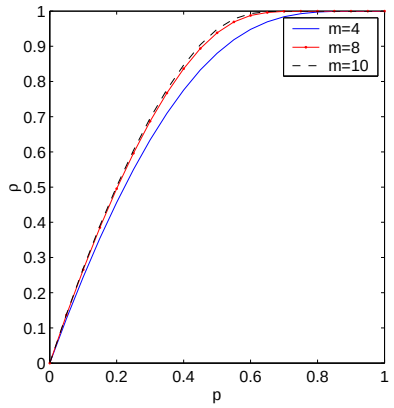


Figure 55: The plot of ρ against p for three different m's ($C = 1$).

Next, we study the convergence of (5.5.50). Fix $m = 8$, for $n = 10, 20, \ldots, 100$, we plot the curve

$$\frac{\hat{E}L_n}{\log_{1/\rho} n},$$

where $\hat{E}L_n$ denotes the estimated expected value of L_n from $10,000$ simulations.

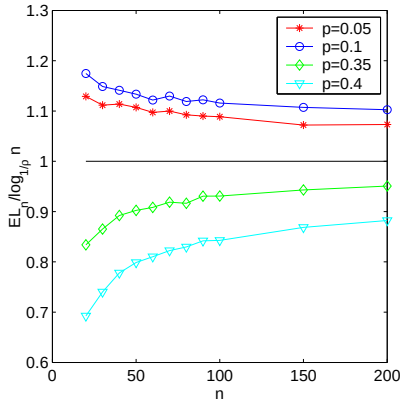


Figure 56: The plot of $\frac{EL_n}{\log_{1/\rho} n}$ against n for two different p's.

From (5.5.50), $EL_n \to \log_{1/\rho} n$. We compare asymptotic EL_n with $\hat{E}L_n$ from simulations in figure 57. Two curves separates as p is above 0.3.

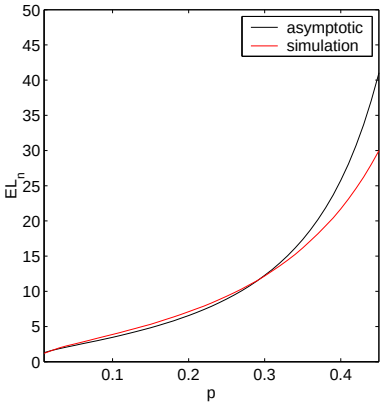


Figure 57: Compare asymptotic EL_n with simulated value against p. ($n = 100$, $m = 8$)

B. Approximation

The approximation formulas (5.5.58) and (5.5.59) provide ways to approximate the distributions of the longest significance run when m or n is large. Figure 58 gives an illustration on the approximation. The left figure shows the approximation for $P(L_0 \geq 8|n = 16, m = 128)$, which requires two simulated probabilities: $P(L_0 \geq 8|n = 16, m = 8)$, and $P(L_0 \geq 8|n = 16, m = 16)$. The right figure shows the approximation for $P(L_0 \geq 8|n = 64, m = 16)$, and it requires two simulated probabilities: $P(L_0 \geq 8|n = 8, m = 16)$, and $P(L_0 \geq 8|n = 16, m = 16)$.

When both m and n are large, we use the approximation formula (5.5.60). The illustration is given in Figure 59. To approximate $P(L_0 \geq 8|n = 64, m = 128)$, the following four simulated probabilities are needed: $P(L_0 \geq 8|n = 8, m = 8)$, $P(L_0 \geq 8|n = 8, m = 16)$, $P(L_0 \geq 8|n = 16, m = 8)$, and $P(L_0 \geq 8|n = 16, m = 16)$.

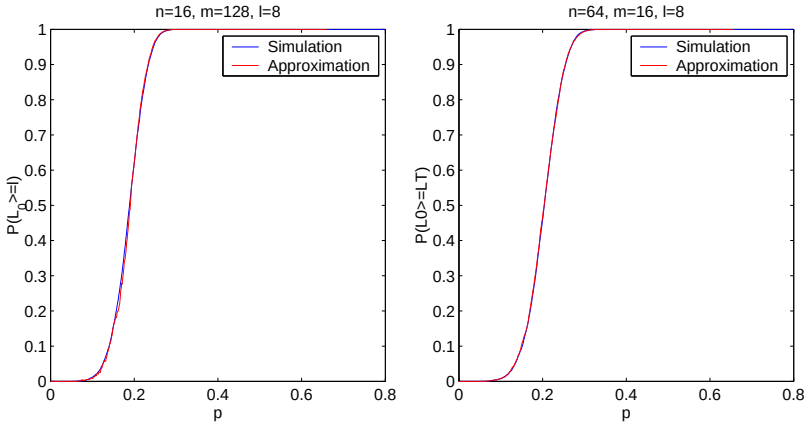


Figure 58: The approximations for large size cases. Left: $n = 16$, $m = 128$, $\ell = 8$; Right: $n = 64$, $m = 16$, $\ell = 8$. Both have $C = 1$.

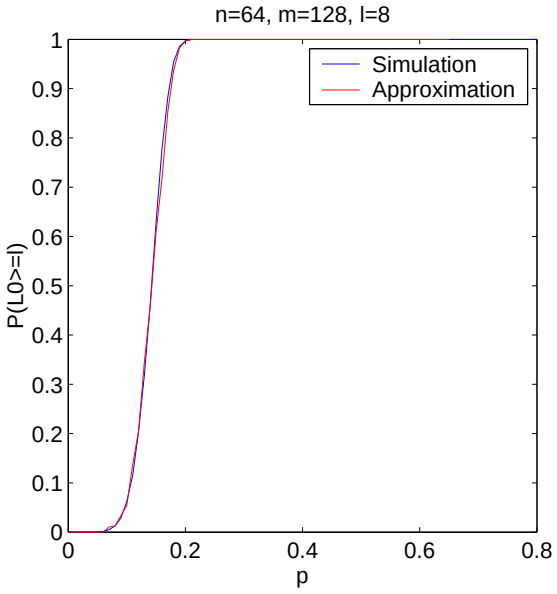


Figure 59: The approximations when both m and n are large. $n = 64$, $m = 128$, $\ell = 8$, $C = 1$.

5.5.3 Hypothesis Testing

We now consider a general hypothesis testing problem for filament detection. Assume on each node of the Bernoulli net, there is a variable $Y_{i,j}$ with cumulative distribution function $F_{i,j}$. We wish to test the null hypothesis of a common cumulative distribution function: $F_{ij} = F_0$ against the alternative that there exists a filament $R(\ell_a)$ such that

$$F_{i,j} = \begin{cases} F_1 & \text{if } (i,j) \in R(\ell_a), \\ F_0 & \text{otherwise,} \end{cases} \tag{5.5.61}$$

where the location of $R(\ell_a)$ is unknown. For a prescribed threshold T, define a Bernoulli variable

$$X_{ij} = \begin{cases} 1 & \text{if } Y_{ij} \geq T, \\ 0 & \text{otherwise.} \end{cases} \tag{5.5.62}$$

The Bernoulli net has success probability $p_{i,j} = 1 - F_{i,j}(T)$. The above problem is equivalent to testing H_0: $p_{ij} = p_0$ against the alternative:

$$H_a: \quad p_{ij} = \begin{cases} p_1 & \text{if } (i,j) \in R(\ell_a), \\ p_0 & \text{otherwise,} \end{cases} \tag{5.5.63}$$

where $p_0 = 1 - F_0(T)$ and $p_1 = \phi(p_0) = 1 - F_1(F_0^{-1}(1 - p_0))$. Assume $p_1 > p_0$.

In general, the length ℓ_a, the location $R(\ell_a)$ and the function $\phi(\cdot)$ are unknown. Given a threshold L_T, the null hypothesis is rejected if $L_0 \geq L_T$.

There are two parameters T(or p_0) and L_T. We consider how to choose the parameters such that the power of the test can be maximized.

5.5.3.1 Simple Alternative

We first consider the simple case of known ℓ_a and $\phi(\cdot)$. The location $R(\ell_a)$ has ignorable effect on the distribution of L_0.

Let $N = (n, m, C)$ be the set of parameters of a Bernoulli net. Let α be the type I error: $\alpha = P(L^* \geq L_T | H_0)$. We write $\alpha = f_N(p, L_T)$ to denote that α is a function of p and L_T. The power of the test depends on p, L_T, ℓ_a and $\phi(\cdot)$. Let power $= g_N(p, L_T, \ell_a, \phi)$. Given significance level α_0, we solve the following problem to obtain the best test,

$$\max_{p, L_T} \quad g_N(p, L_T, \ell_a, \phi) \tag{5.5.64}$$

$$s.t \quad f_N(p, L_T) \leq \alpha_0, \tag{5.5.65}$$

We first solve for (5.5.65). Define

$$p_N^*(\alpha_0, L_T) = \sup\{p : f_N(p, L_T) \leq \alpha_0\}. \tag{5.5.66}$$

Later we will write $p^*(L_T)$ for simplicity. There are a group of α_0-points: $(L_T, p^*(L_T)), L_T = 1, \cdots, n$. Figure (60) gives an illustration of the change of type 1 error versus p for different L_T's. In (5.5.64), the optimization problem is to solve

$$g_N^*(\ell_a, \phi) = \max_{L_T} \quad g_N(p^*(L_T), L_T, \ell_a, \phi). \tag{5.5.67}$$

Denote the best group as $(L_T^*, p^*(L_T^*))$.

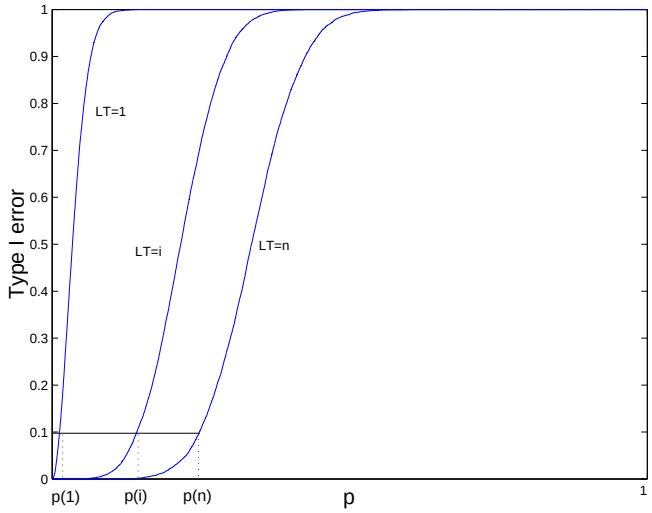


Figure 60: An illustration of $f(\cdot, L_T)$ curves. The points are marked for $\alpha_0 = 0.1$.

In the following, we provide a simple approximation for the power of the test. We denote $L_n(p)$ the longest significance run in one dimension when the length of the Bernoulli sequence is n and the success probability is p. For simplicity, we will write the null hypothesis as $H_0 : \ell_a = 0, p_0$ and the alternative $H_a : \ell_a = \ell_0, p_1 = \phi(p_0)$.

Let $R(\ell_0)$ be the filament of length ℓ_0. Denote E the event that there is a significance run longer or equal to ℓ on $R(\ell_0)$. Let E' be the complement of E. Then

$$\begin{aligned} P(L_0 \geq \ell) &= P(L_0 \geq \ell|E)P(E) + P(L_0 \geq \ell|E')P(E') \\ &= [1 - P(L_0 \geq \ell|E')]P(E) + P(L_0 \geq \ell|E') \end{aligned}$$

Under H_a, without loss of generality, suppose on the filament $R(\ell_0)$ has success probability p_1. Based on the following approximations,

$$P_{H_a}(L_0 \geq \ell|E') \approx P_{H_0}(L_0 \geq \ell|E') \approx \alpha_0,$$

we derive a simple approximation for the power of the test.

$$P_{H_a}(L_0 \geq \ell) \approx \alpha_0 + P[L_{\ell_0}(p_1) \geq \ell] - P[L_{\ell_0}(p_0) \geq \ell] \tag{5.5.68}$$

Examples: For fixed type I error $\alpha_0 = 0.05$, we plot the curves of $p*(L_T)$ against L_T for various m and n's (Figure 61).

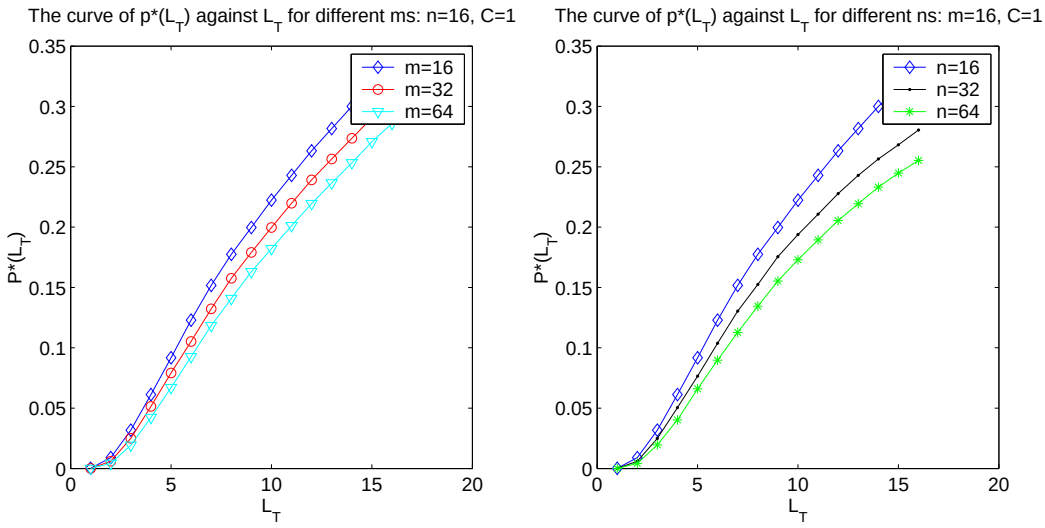


Figure 61: The plot of $p^*(L_T)$ against L_T. Left: the change of the curve with various m's ($n = 16$, $C = 1$); Right: the change of the curve with various n's ($m = 16$, $C = 1$).

5.5.3.2 Composite Alternatives

More generally, Let H_a be an element of a well-defined set Ω. For significant level α, the power is

$$\text{power}(L_T, p) = \int_\Omega Pr[L^* \geq L_T|H_a]dP(H_a). \tag{5.5.69}$$

Assume $\ell_a \in \Lambda$ and $\phi(\cdot) \in \mathcal{F}$. Then

$$\Omega = \{(\ell_a, \phi) : \ell_a \in \Lambda, \phi \in \mathcal{F}\}.$$

$$\text{power}(L_T, p) = \int_{\phi \in \mathcal{F}} \{ \sum_{\ell_a \in \Lambda} Pr(L_0 \geq L_T | \ell_a, \phi) P(\ell_a) \} dP(\phi). \tag{5.5.70}$$

Let $\mu_\ell = \int_{\ell_a \in \Lambda} \ell_a dP(\ell_a)$ and $\mu_\phi = \int_{\phi \in \mathcal{F}} \phi dP(\phi)$.

$$\text{power}(L_T) \approx Pr(L^* \geq L_T | \mu_\ell, \mu_\phi). \tag{5.5.71}$$

5.5.3.3 An Example

Next, we show a simple example to illustrate the procedures for parameter selection.

Example 1: Consider the hypothesis test: $H_0 : Y_{ij} \sim N(0,1)$ against $H_a : Y_{ij} \sim N(\theta, 1), (i,j) \in R(\ell_a)$.

The first step is to find $p^*(L_T)$ for a given α_0 and L_T.

The second step is to simulate the power given L_T and $\phi[p^*(L_T)]$. The function $\phi(\cdot)$ is calculated as follows. Given T, we have $p_0 = 1 - \Phi(T)$ and

$$p_1 = \phi(p_0) = 1 - \Phi(T - \theta) = 1 - \Phi(\Phi^{-1}(1 - p_0) - \theta),$$

where Φ is the cdf of normal distribution.

We thus obtain the power values for different L_T's, and the optimal power can be found.

5.5.4 Conclusions

In this section, we study some asymptotic properties of the distribution of the longest run in a Bernoulli net. A simulation approach to find the optimal power is provided. Theoretical analysis will be an interesting future work.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

This thesis studies the applications of sparse modeling in the areas of classification, compression and detection. This chapter gives some conclusions and possible future work.

In the first part, we incorporate feature sparsity to improve the performance of support vector machines. For hyperplanes refined to certain feature subspace, we derive a bound on the VC dimension that is proportional to the dimensionality of the subspace. This is the motivation of our new approach Penalized Support Vector Machines. An analysis of the dual leads to the penalized least square problem. We discuss the statistical properties following the line of [27]. Our approach trains SVM while doing feature selection automatically and simultaneously. However, a main shortcoming of our approach is that one more parameter is included to optimize compared to SVMs. Further research can be continued on how to adjust the GACV method [67] to tune the trade-off parameters. A challenging topic for future research is to study whether the proposed approach perform as well as the oracle procedure for feature selection, a property studied in [27]. Moreover, the current feature selection approach can only work with linear kernel and the generalization to nonlinear kernels will be very interesting.

In the second part, we studied three heuristics in building a cascade. The three heuristics are (1) frontier following approximation, (2) controlling error rates, and (3) weighting. Simulations on synthetic data are carried out. It is found that weighting heuristic is optimal in both computational complexity and error rates. The current winner – weighting algorithms – is applied to an IR data set. It is found that it outperforms some state-of-the-art approaches that utilize the same type of simple classifiers. However, it fails to some classifiers that utilize more complex decision rules. It will be interesting to examine how to integrate the cascade algorithm with other decision rules other than univariate classifier. Another interesting problem is to study whether the regularization (or the penalized likelihood methods) can be a framework to build a cascade model. If yes, what is(are) the advantage(s)?

In the third part, utilizing a multiscale structure for line segments – beamlet – a zero-tree like coding method is designed for generic curves. The designed method is progressive and easy to implement. Numerical experiments demonstrate an advantage over existing curve coding software – JBIG. A more important purpose of this work is to describe in detail how to coding single line segments (single beamlets) and how to unify them through zero-tree coding, so that any curve can be coded. The overall approach is novel and has not

been studied well in the literature. So far representation is separated from coding. In the problem of wavelet coding, the problem is more straightforward; hence the rate-distortion consideration is integrated with the zero-tree coding. It is not clear how such an integration can be achieved in a beamlet based coding. The challenge is that in a beamlet coding, when distortion changes, the geometry (which is represented by the structure of the quad-tree, as recorded in (23) and (24)) can be changed as well; while in wavelet coding, the quad-tree structure is fixed. Making the coder a more coherent one is an interesting future research.

The fourth part is about the detection of the presence of embedded objects. Significance Run Algorithm (SRA) serves as a foundation to organize the underlying objects, and has been applied successfully for the detection in noisy images and homogeneous texture background. The distribution of the longest significance run in a Bernoulli net is a challenging problem in mathematics, and the research is still ongoing. We provide a simulation approach for parameter selection to optimize the power of the test. Some interesting observations were presented, and we are still unable to prove them in a rigorous way. We will also explore how to use the interdependence in the Bernoulli net to improve the power of the test.

REFERENCES

[1] *Data Web Site*. `http://www.ee.columbia.edu/~luoht/research/rvSeg/`.

[2] *Jbeam Software*. `http://www.isye.gatech.edu/~xiaoming/jbeam2004/`.

[3] Aghito, S. M. and Forchhammer, S., "Contex based coding of binary shapes by object boundary straightness analysis," *Proceedings of Data Compression Conference*, pp. 399–408, March 2004.

[4] Ahuja, R., Magnanti, T., and Orlin, J., *Network Flows: Theory, Algorithms, and Applications*. New Jersey: Prentice Hall, 1993.

[5] Arias-Castro, E., Donoho, D. L., and Huo, X., "Adaptive multiscale detection of filamentary structures embedded in a background of uniform random points," tech. rep., Stanford University, `http://www-stat.stanford.edu/~donoho/reports.html`, 2003.

[6] Arias-Castro, E., Donoho, D. L., and Huo, X., "Asymptotically optimal detection of geometric objects by fast multiscale methods," tech. rep., Stanford University, `http://www-stat.stanford.edu/~donoho/reports.html`, 2003.

[7] Arias-Castro, E., Donoho, D. L., Huo, X., and Tovey, C., "Connect-the-dots: How many random points can a regular curve pass through?," tech. rep., Stanford University, `http://www-stat.stanford.edu/~donoho/reports.html`, 2003.

[8] Averbuch, A., Coifman, R. R., Donoho, D. L., Israeli, M., and Walden, J., "Fast slant stack: A notion of radon transform for data in a cartesian grid which is rapidly computable, algebraically exact, geometrically faithful and invertible," *Technical Report*, 2001.

[9] Balakrishnan, N. and Koutras, M. V., *Runs and Scans with Applications*. New York: John Wiley, 2nd ed., 2002.

[10] Bennett, K. P., Cristianini, N., Shawe-Taylor, J., and Wu, D., "Enlarging the margins in perceptron decision trees," *Machine Learning*, vol. 41, pp. 295–313, December 2000.

[11] Biswas, S., "Contour coding through stretching of discrete circular arcs by affine transformation," *Pattern Recognition*, vol. 34, pp. 63–77, 2001.

[12] Breiman, L., "Heuristics of instability and stabilization in model selection," *The Annals of Statistics*, vol. 10, pp. 2350–2383, September 1996.

[13] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J., *Classification and regression trees*. Belmont, CA: Wadsworth Advanced Books and Software, 1984.

[14] Burges, C., "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 955–974, 1998.

[15] Burr, E. J. and Cane, G., "Longest run of consecutive observations having a specified attribute," *Biometrika*, vol. 48, 1961.

[16] Burt, P. J. and Adelson, E. H., "The laplacian pyramid as a compact image code," *IEEE Transactions on Communications*, vol. 9, no. 4, pp. 532–540, 1983.

[17] Chapelle, O., Vapnik, V., Bousquet, O., and Mukherjee, S., "Choosing multiple parameters for support vector machines," *Machine Learning*, vol. 46, no. 1, pp. 131–159, 2002.

[18] Cristianini, N. and Shawe-Taylor, J., *Support vector machines and other kernel-based learning methods*. New York: Cambridge University Press, 2000.

[19] Darling, R. W. R. and Waterman, M. S., "Extreme value distribution for the largest cube in a random lattics," *SIAM J. Appl. Math.*, vol. 46, 1986.

[20] Do, M. N., Dragotti, P. L., Shukla, R., and Vetterli, M., "On compression of two-dimensional piecewise smooth functions," *Proc. of IEEE International Conference on Image Processing (ICIP)*, October 2001.

[21] Donoho, D. L., "Wedgelets: nearly minimax estimation of edges," *Annals of Statistics*, vol. 27, pp. 859–897, 1999.

[22] Donoho, D. and Huo, X., "Beamlets and multiscale image analysis," in *Multiscale and Multiresolution Methods*, vol. 20 of *Lecture Notes in Computational Science and Engineering*, Springer, 2001.

[23] Elad, M., Hel-Or, Y., and Keshet, R., "Pattern detection using a maximal rejection classifier," *Pattern Recognition Letters*, vol. 23, pp. 1459–1471, Octorber 1983.

[24] Erdos, P. and Renyi, A., "On a new law of large number," *Journal Anal. Math.*, vol. 22, 1970.

[25] Erdos, P. and Revesz, P., "On the length of the lonest head run," *Colloq. Math. Soc. Janos Bolyai,* **16, Topics in Information Theory**, 1975.

[26] Etoh, M., Ostermann, J., and Sikora, T., "Editorial: special issue on shape coding for emerging multimedia applications," *Signal Processing: Image Communication*, vol. 15, 2000.

[27] Fan, J. and Li, R., "Variable selection via nonconcave penalized likelihood and its oracle properties," *Journal of the American Statistical Association*, vol. 96, pp. 1348–1360, 2001.

[28] Freeman, H., "On the coding of arbitrary geometric configurations," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 260–268, June 1961.

[29] Freeman, H., "Application of the generalized chain coding scheme to map data processing," *Proc. IEEE Comput. Soc. Conf. Pattern Recog. Image Processing*, pp. 220–226, May 1978.

[30] Friedman, J. H., "Greedy function approximation: a gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[31] Friedman, J. H., "Tutorial: Getting started with mart in r," *http://www-stat.stanford.edu/∼jhf/*, April 2002.

[32] Friedman, J. H., Hastie, T., and Tibshirani, R., "Additive logistic regression: a statistical view of boosting," *The Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[33] Gibbons, J., *Nonparametric Statistical Inference*. New York: MCGraw-Hill, 1971.

[34] Hastie, T., Friedman, J., and Tibshirani, R., *Elements of Statistical Learning: data mining, inference and prediction*. New York: Springer-Verlag, 2001.

[35] Hastie, T., Tibshirani, R., and Friedman, J., *The Elements of Statistical Learning*. New York: Springer, 2001.

[36] Hearn, D. and Baker, M., *Computer Graphics: C Version*. Prentice Hall, 2nd ed., 1996.

[37] Heisele, B., Serre, T., Mukherjee, S., and Poggio, T., "Feature reduction and hierarchy of classifiers for fast object detection in video images," *Proceedings of 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 18–24, December 2001.

[38] Hel-Or, Y. and Hel-Or, H., "Real time pattern matching using projection kernels," *The 9th IEEE International Conference on Computer Vision*, October 2002.

[39] Hel-Or, Y. and Hel-Or, H., "Generalized pattern matching using orbit decomposition," *IEEE International Conference on Image Processing*, vol. 3, pp. 69–72, 2003.

[40] Howard, P. G. and et. al., "The emerging jbig2 standard," *IEEE Trans. on Circuits and Sys. for Video Tech.*, vol. 8, pp. 838–848, November 1998.

[41] Huo, X., "A geodesic distance and local smoothing based clustering algorithm to utilize embedded geometric structures in high dimensional noisy data," *SIAM International Conference on Data Mining, Workshop on Clustering High Dimensional Data and its Applications*, May 2003.

[42] Huo, X., Chen, J., and Donoho, D., "Multiscale significance run: Realizing the 'most powerful' detection in noisy images," *Asilomar Conference on Signals, Systems, and Computers*, November 2003.

[43] Huo, X., Chen, J., and Donoho, D., "Multiscale detection of filamentary features in image data," *SPIE Wavelet-X*, August 2003.

[44] Huo, X. and Donoho, D., "Local linear projection (llp)," *First IEEE Workshop on Genomic Signal Processing and Statistics (GENSIPS)*, October 2002.

[45] Katsaggelos, A., Kondi, L., F.W. Meier, J. O., and Schuster, G., "Mpeg-4 and rate-distortion-based shape-coding techniques," *Proceedings of the IEEE*, vol. 86, pp. 1126–1154, June 1998.

[46] Kim, J., Bovik, A., and Evans, B., "Generalized predictive binary shape coding using polygon approximation," *Signal processing: Image Communication*, vol. 15, pp. 643–663, 2000.

[47] Kohavi, R. and John, G. H., "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 1–2, pp. 273–324, 1997.

[48] Koplowitz, J. and DeLeone, J., "Hierarchical representation of chain-encoded binary image contours," *Computer Vision and Image Understanding*, vol. 63, no. 2, pp. 344–352, 1996.

[49] Lu, C. C. and Dunham, J. G., "Highly efficient coding schemes for contour lines based on chain code representation," *IEEE Trans. Communications*, vol. 39, pp. 1511–1514, October 1991.

[50] Meer, P., Sher, A., and Rosenfeld, A., "The chain pyramid: hierarchical contour processing," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, pp. 363–376, April 1990.

[51] Murphy, P. M. and Aha, D. W., "Uci repository of machine learning databases," tech. rep., Department of Information and Computer Science, University of California, Irvine, `http://www.ics.uci.edu/~mlearn/MLRepository.html`, 1992.

[52] Osuna, E., Freund, R., and Girosi, F., "Support vector machines: Training and applications," Tech. Rep. AIM-1602, MIT A. I. Lab, 1996.

[53] Papastavridis, S. G. and Koutras, M. V., "Bounds for reliability of consecutive-k-within-m-out-of-n systems," *IEEE Trans. Reliab.*, vol. 42, 1993.

[54] Petrov, V. V., "On the probabilities of large deviations for sums of independent random variables," *Theory Prob. Appl.*, vol. 10, 1965.

[55] Quinlan, J. R., *C4.5: Programs for machine learning*. Morgan Kaufmann, 1993.

[56] Roweis, S. and Saul, L., "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, 2000.

[57] Said, A. and Pearlman, W., "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 243–250, June 1996.

[58] Schapire, R., Freund, Y., Bartlett, P., and Lee, W. S., "Boosting the margin: a new explanation for the effectiveness of voting methods," *The Annals of Statistics*, vol. 26, no. 5, pp. 1651–1686, 1998.

[59] Schuster, G. and Katsaggelos, A., "An optimal polygonal boundary encoding scheme in the rate distortion sense," *IEEE Trans. Image Processing*, vol. 7, pp. 13–26, January 1998.

[60] Shapiro, J., "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Transactions on Signal Processing*, vol. 41, pp. 3445–3462, December 1993.

[61] Shukla, R., Dragotti, P. L., Do, M. N., and Vetterli, M., "Rate-distortion optimized tree structured compression algorithms for piecewise smooth images," *Submitted to IEEE Transactions Image Processing*, 2002.

[62] Tenenbaum, J., Silva, V., and Langford, J., "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, 2000.

[63] Ulichney, R., "Bresenham-style scaling," *Proceedings of the IS&T Annual Conference*, pp. 101–103, 1993.

[64] Vapnik, V., *The Nature of Statistical Learning Theory*. New York: Springer Verlag, 1995.

[65] Vapnik, V. and Chervonenkis, A., "On the uniform convergence of relative frequencies of events to their probabilities," *Theory of Probability and Its Applications*, vol. 16, pp. 264–280, 1971.

[66] Viola, P. and Jones, M., "Robust real-time object detection," *ICCV Workshop on Statistical and Computation Theories of Vision*, July 2001.

[67] Wahba, G., Lin, Y., and Zhang, H., "Gacv for support vector machines," in *Advances in Large Margin Classifiers* (Smola, A., Bartlett, P., Scholkopf, B., and Schuurmans, D., eds.), pp. 297–311, New York: MIT Press, 2000.

[68] Wallenstein, S., Nau, J., and Glaz, J., "Power of the scan statistic in detecting a changed segment in a bernoulli sequence," *Biometrika*, vol. 81, 1994.

[69] Wang, H., Schuster, G. M., Katsaggelos, A. K., and Pappas, T. N., "An efficient rate-distortion optimal shape coding approach utilizing a skeleton-based decomposition," *IEEE Trans. Image Processing*, vol. 12, pp. 1181–1193, October 2003.

[70] Weston, J., Mukherjee, S., Chaperlle, O., Pontil, M., Poggio, T., and Vapnik, V., "Feature selection for svms," *Advances in Neural Information Processing Systems*, vol. 13, 2000.

[71] Witten, I., Neal, R., and Cleary, J., "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, June 1987.

[72] Zhu, M. and Hastie, T., "Feature extraction for non-parametric discriminant analysis." Submitted.